2010/202

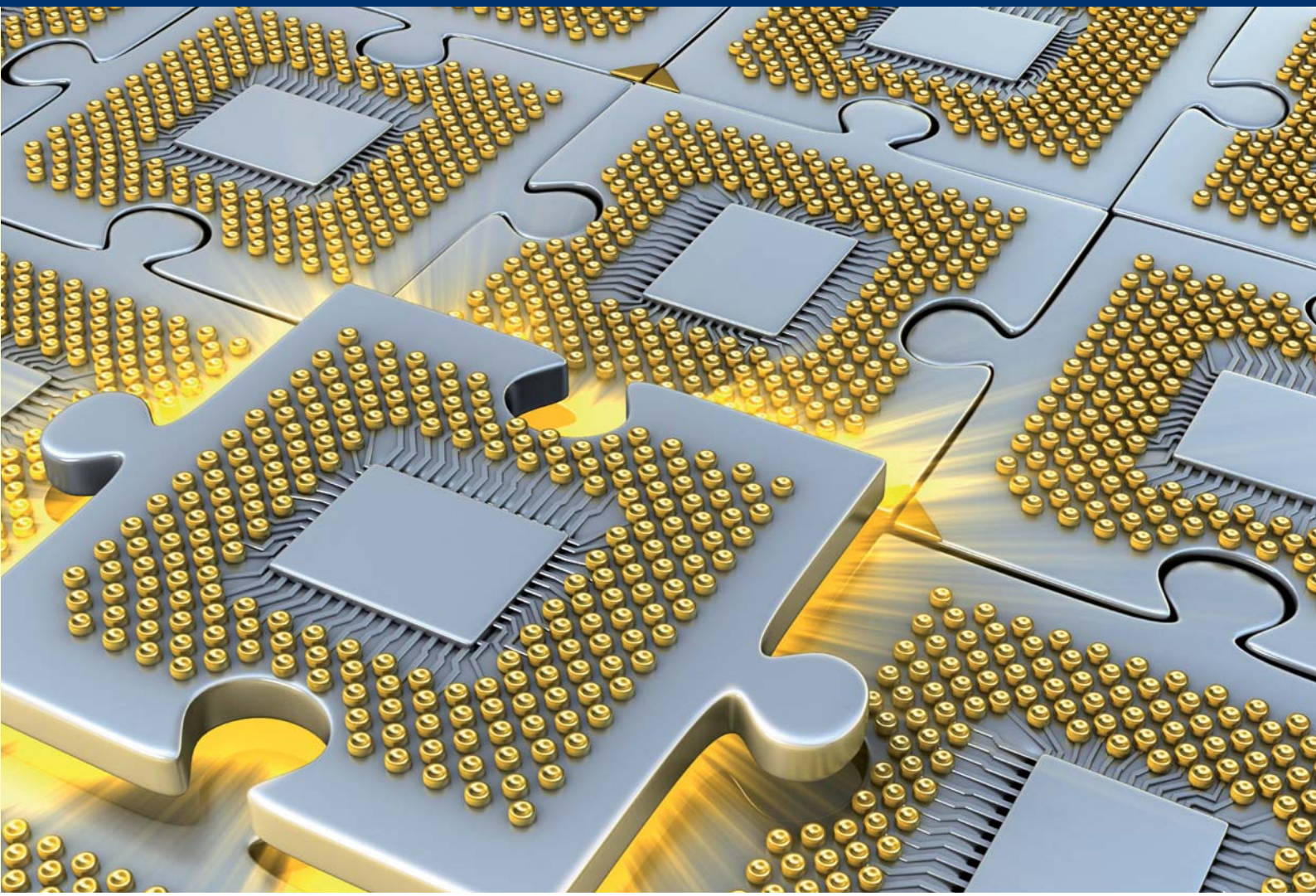
Fraunhofer IAO

FRAUNHOFER-INSTITUT FÜR ARBEITSWIRTSCHAFT UND ORGANISATION IAO

Dieter Spath, Anette Weisbecker (Eds.), Erik Hebisch

MARKET OVERVIEW OF TOOLS FOR MULTICORE SOFTWARE DEVELOPMENT

MWARE: SOFTWARE TECHNOLOGIES FOR THE MULTICORE FUTURE



FRAUNHOFER VERLAG

MWARE multicore software

FRAUNHOFER-INSTITUT FÜR ARBEITSWIRTSCHAFT UND ORGANISATION IAO

Dieter Spath, Anette Weisbecker (Eds.), Erik Hebisch

MARKET OVERVIEW OF TOOLS FOR MULTICORE SOFTWARE DEVELOPMENT

MWARE: SOFTWARE TECHNOLOGIES FOR THE MULTICORE FUTURE

FRAUNHOFER VERLAG

Contact
Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO
Nobelstraße 12, 70569 Stuttgart
www.mware.fraunhofer.de/EN/

Editors
Univ.-Prof. Dr.-Ing. Dr.-Ing. E.h. Dieter Spath
Priv. Doz. Dr.-Ing. habil. Anette Weisbecker

Authors
Dipl.-Inform. Erik Hebisch

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliografic data is available in the Internet at <http://dnb.d-nb.de>.
ISBN: 978-3-8396-0165-5

Printing and Bindery
Mediendienstleistungen des
Fraunhofer-Informationszentrum Raum und Bau IRB, Stuttgart

Printed on acid-free and chlorine-free bleached paper.

Publisher
FRAUNHOFER VERLAG
Fraunhofer Information-Centre for Regional Planning and Building Construction IRB
P.O. Box 80 04 69, D-70504 Stuttgart
Nobelstrasse 12, D-70569 Stuttgart
Phone: +49 (0) 711 970-2500
Fax: +49 (0) 711 970-2508
E-Mail verlag@fraunhofer.de
URL: http://verlag.fraunhofer.de

Preface

Software developers have long profited from the advances in processor technology. With new processors generally being faster and more complex existing programs were executed faster without being modified. Software developers used the extra computational power to implement more features in their programs and thus increasing the attractiveness of their products.

While the trend of increased complexity in processors has not been interrupted it is unlikely that processors speed can be increased further. Instead, multicore processors promise a further increase in computational power by fitting multiple cores on one chip and using those to do calculations in parallel. Aside from the benefit of parallel computation, this design is cost- and energy-efficient – a perfect fit for the future of mobile and ubiquitous computing.

But software developers are facing a new challenge. No longer do they get speed improvements for free; they have to parallelize their software in order to take advantage of multiple cores. They have to figure out which parts of their programs can be executed in parallel and how to avoid errors while implementing a parallel version.

The goal of the MWare project, funded by the Fraunhofer Gesellschaft, is to research tools, methods and processes for multicore software development. This research covers the parallelization of existing software where tools are needed to analyze the compute intensive parts that will benefit from parallelization, to debug the software after the parallelization and to tune the parallel version for optimal performance. Furthermore, tools are needed for implementing parallel software, such as libraries implementing parallel algorithms, languages that offer built-in parallel semantics and modeling tools for designing with parallelism in mind.

The field of software engineering has begun to regard parallelism as an important part of the future of software development. We hope that this study will provide you with useful information on the tools that ease the transition into this new mindset.

The Editors

MWARE multicore software

Table of Contents

List of Figures

List of Tables

1 Introduction

The well-known Moore's law states that the number of transistors on a chip doubles every two years. More transistors mean more computing power. Combined with increasing clock speed this trend is responsible for the powerful processors we use today.

The trend is still accurate. However the clock speed of current processors has stagnated in the last years. Chip manufactures cannot increase the speed any further without risking or sacrificing power efficiency. Higher processor clock speeds require more power. But faster processors also generate more heat and thus require even more power for cooling. The result is that the ratio of compute power per watt doesn't scale linearly. This is why experts speak of single-core processors hitting the »power wall«, where the amount of additional compute power does not justify the additional amount of actual power that is required to achieve it.

However speed was not the only improvement in the field of processor development. New chip production techniques achieved a constant miniaturization of chip components which means that more processors can be placed on the same area. This is one of the reasons for the current developments in the field of multicore CPUs.

Instead of making single processors faster, more, smaller processors are placed on the same area. Instead of boosting the processing power by increasing clock speeds, additional processing power is achieved by using multiple processors simultaneously. This has significant consequences for software developers. Until now they could take it for granted that the improvements in processor technology will in turn directly improve their software's speed. Furthermore, the single processor model has offered them a deterministic computational model that was straight-forward to understand and to debug in the case of errors.

Now they have to adapt their software to the hardware and learn how to distribute the workload on multiple processors in parallel. They have to do so in order to implement new features in their software without degrading performance.

This study provides an overview of the tools and their requirements of software development for multicore CPUs.

1.1 Areas of application

Parallel and distributed computing is common practice in scientific computing and has made possible many of the scientific discoveries in the fields of medicine, physics, engineering and many more. In order to cope with the rising need for processing power researchers created compute grids by connecting many single computers in a network and distributing computational work to them. This way they can simulate multiple different scenarios at once or single steps of one simulation in parallel. Today's super computers feature an impressive amount of cores and memory capacity. The distribution of work is central to the parallel processing paradigm.

The knowledge on parallel distributed computing accumulated by the scientific computing community forms the basis of the programming abstractions for today's programmers that develop software for desktop multicore CPUs.

While today's commodity hardware doesn't quite have the amount of cores of a Top500 super computer, quad-core and octa-core processors are becoming more and more commonplace in desktop computers and laptops. Furthermore, with techniques like Intel's Hyperthreading, each of the physical processor cores may offer additional, hardware scheduled threads, thus increasing the amount of possible parallelism even more.

Miniaturization and reduced energy consumption are the driving factors behind multicore adoption in smaller electronics devices like netbooks or smartphones. These devices are expected to do more than ever on a single battery charge. In this regard smaller processors mean more space for a bigger battery and less power-hungry processors mean that this battery will drain slower. Of course this win-win situation relies on developers implementing software that uses the available cores efficiently.

1.2 Software

Why do multicore processors have an impact on software development? What exactly has changed?

Two things have changed. First, the speed of single processors has stagnated and second, multiple processor cores are now common even in consumer hardware. One thing that hasn't changed is the demand for software with powerful features and excellent performance.

Software developers have profited from the advancements in processor technology for a long time. Software automatically got faster due to improved clock

speed and advanced processor features such as instruction pre-fetching and branch prediction. Now that the processor speeds have leveled, developers have to modify their software to take advantage of the new hardware.

In order to maintain or improve performance, developers have to use more cores in parallel. This means that they have to restructure their software to allow for parallel execution. They have to decide which operations to parallelize, how to implement the parallelism and how to deal with the new challenges that arise with managing parallelism and concurrency.

1.3 Concurrency and parallelism

Concurrency and parallelism are closely related. The term concurrency describes a system in which more than one process is executed at a time. Parallelism is the simultaneous computation of many calculations. The difference is subtle. Concurrency is a system characteristic and parallelism is a form of computation.

All operating systems since the mid-nineties are concurrent, i.e. they can run multiple programs at once, a fact that is advertised as »multi-tasking«. With only one processor, the simultaneous execution of multiple programs is an illusion. The illusion is achieved by letting one program use the processor exclusively for a certain amount of time, then suspending it and switching to another program. The program state is saved before switching and restored before letting it run again. The state comprises the values of the different registers at the time, the runtime stack and the program counter that points to the next operation to be computed when the execution is resumed. The execution and state management is done by the operating system's scheduler. The time slice for every program is so small that the user never notices the constant switching of instruction streams happening in the scheduler. Today's operating systems still have to do this switching since the number of concurrently running programs is higher than the number of available processor cores.

The smallest unit of processing that is recognized by the operating system's scheduler is the thread of execution. A single-threaded program is run in a process that consists of only one thread. Every operating system offers a way for creating additional threads from within a program. These additional threads run in the same process but are scheduled along with all the other running threads from all other processes in the system. The programmer creates a thread from an existing function. An application which uses threads is called multi-threaded.

Applications that benefit from multi-threading are those with a graphical user interface. While performing a long calculation the user interface has to be re-

drawn to reflect the state of the calculation. Without multi-threading, the calculation will block the redraw and the user interface will appear frozen, unable to accept and react to user input. There are ways around this without threading, but the usage of threads reflects best what the application should accomplish: Giving feedback on the calculation while the calculation is running.

Concurrent execution provided by the operating system is inherently nondeterministic, at least for the developer. In order to maintain concurrency the operating system's scheduler can suspend a thread at any time and switch to another. This does not pose any problems for programs with only a single thread; they simply continue from the moment they were interrupted and are guaranteed that no other program has modified their internal state. But when running multiple threads in the same program the developer has to verify this fact for himself. This is because when a thread is active it can access all the shared data structures in the program, even if another thread of the same program is busy manipulating them. The programmer has to make sure that the two manipulations don't result in a data corruption, e.g. when one thread changes a data structure the other one is depending on. The developer has to make sure to consider all possible interactions of the threads he uses in order to protect the right data, which becomes non-trivial as the number of threads, and thus the number of possible interactions, increases.

With multicore processors, this challenge has become even more difficult. Multicore processor can perform multiple calculations at once, each on its own core. Concurrent execution is no longer an illusion. The operating system schedules the threads on the different cores and the likelihood of two threads of the same program running at exactly the same time, and therefore possibly interfering with each other, has increased.

Another issue that arises with parallelization is the efficient utilization of multiple cores. A parallel program should distribute its workload evenly to the available processor cores in order to decrease the time spent for computation. This is however not always possible as the workload might not be evenly dividable or there are other programs running that want to get their share of the processor's time. In a complex system such as an operating system it is not always clear how to parallelize a piece of software to achieve maximum performance on the one hand without risking oversubscribing processor time and thus impeding system performance on the other hand.

These are the new challenges that software development has to deal with in the multicore era.

2 Multicore Software Engineering

In order to tackle the challenges of parallel programming in the multicore era, the software development process has to consider the identification, coordination, implementation and verification of parallel operations in software projects. These activities are described in the next paragraphs.

2.1 Identification

The identification of parallelizable operations is a fundamental activity when developing multicore software. It can be guided by performance or architectural considerations.

If the goal is to speed up a given software it has to be analyzed where it spends the most time during its execution. This is done by examining the software during runtime and recording which functions get called, how long they take to finish, which resources are used and so on. For the longest running parts, i.e. where a speed up would significantly improve the overall runtime, it has to be determined if they can be split up in way that allows for parallel execution. Long running operations that modify the contents of an array are typical candidates for parallelization if the individual elements can be modified independently from another. Otherwise, the dependencies must be resolved in order to maintain correctness. Furthermore, the access of shared resources like shared data structures, cache memory, input and output devices has to be observed to find additional dependencies between operations.

The identification of parallelizable operations can also be done on a conceptual level during the design phase of a software. Here, the operations and the data they operate on are specified in a way that makes parallelization possibilities obvious. The goal is to identify certain patterns in the structure of the software that will guide the developers to an efficient parallel implementation.

The two approaches are not mutually exclusive and can be used during the modification of an existing software or the development of a new software.

2.2 Coordination

Coordination is needed to control the different threads and their possible interactions during execution. This is where the identification of dependencies is important. An operation may depend on the result of another operation. It has to wait for the result thus creating a sequential dependency. This dependency can sometimes be avoided by restructuring the algorithm to allow for computa-

tion of sub results that can later be combined. A good example is the map/reduce parallel pattern. The original problem is partitioned and for each partition the operation calculates a sub result. Speedup is achieved by running multiple operations that operate on different sub results simultaneously. All the sub results are collected at the end and reduced to the final result.

Other dependencies are created by accessing shared resources. These can be output devices, e.g. writing a message to a log file. If the access is uncontrolled then the messages of two parallel operations can interleave which is often contrary to the developer's intention. Another example of a shared resource is an array that is visible to all functions in the software. If the access is uncoordinated it may happen that while one operation is iterating over the array to compute something, another operation is modifying it at the same time which will result in false results or even software crashes if the second operations deletes elements of the array the first operations needs. This kind of dependency can be removed by duplicating the shared resource, e.g. by making a copy of the array before iterating over it. This is however not possible with all resources, e.g. the log file from the example above cannot usually be duplicated. The access has to be coordinated, which is usually done by attempting to lock the resource while operating on it. This guarantees that the operation which acquires the lock is the only one operating on it; the others have to wait their turn. A problem arises when an operation needs more than one shared resource. Then it may happen that it can lock the first resource but fails to lock the second one and has to wait. If another operation holds the lock on the second resource and then tries to lock the first one a deadlock is created. Since both operations wait for the other one to give up the shared resource while holding the resource the other one is waiting for, they both cannot proceed any further.

The coordination of the parallel execution becomes more difficult with a higher number of concurrent operations. The developer has to think about the possible interleaving of the operations and coordinate the access to shared resources accordingly. Specifying the operations and their data dependencies gives information about the temporal dependencies of the individual operations. By specifying the pre- and post-condition of the operations an execution model of the program flow can be derived that can serve as the basis for automatic error checking or as a guide to the developer trying to verify the design.

2.3 Implementation

The choice of an implementation strategy is guided by the identification of the parallelizable parts and their dependencies. The goal is to develop a solution that is both correct from a coordination standpoint and efficient from a performance standpoint.

The easiest way is to use an existing library that offers a parallel version of the algorithm one is using. Coordination and efficiency aspects are handled by the library; the developer only has to understand how to use its interface. If there is no ready-to-use library, parallel patterns give guidance for the most common patterns found when implementing parallel algorithms. The parallel patterns can be formalized in programming frameworks where a developer only has to plug-in the code that is specific to his application. The framework's runtime coordinates the execution.

Without tailor-made parallel solutions, the developer has to develop a parallel solution himself. He can use thread libraries to manually create threads and schedule operations to run in parallel or he can use libraries which offer higher-level abstractions like actors or dataflow concepts to express parallelism with. These higher-level libraries impose some restrictions on how data is accessed and exchanged in order to make the coordination aspects of parallelization easier or hide them from the developer.

High-level concepts also form the basis for programming languages that are especially suited for parallel computation. Concepts like actors, message passing or dataflow are built into the language. With these concepts built-in and used by the developer the compiler or the runtime system can enforce and assume certain properties of the code and optimize accordingly.

2.4 Verification

After implementing a parallel software it has to be verified that it runs correctly, produces correct results and does so efficiently.

The program flow of a parallel application can be non-intuitive due to the parallel execution of multiple operations at once. While the coordination strategy should prevent common bugs such as the concurrent modification of shared data structures understanding the flow of execution is non-trivial. Depending on the kind of implementation that was chosen reproducing the actual interactions between the different operations can be difficult. When using threads the scheduling is nondeterministic and thus the interleaving can differ each time the program is executed. If the developer uses a high-level library or language he has to use tools that provide him with useful high-level analytics so that he can understand and correct possible errors.

Software development for multicore architectures has to concern itself also with the efficient utilization of hardware. The software should make use of the available processors and should ideally scale automatically to more processors in the future. This way, it can once again directly profit from the improvements in processor technology.

3 Multicore Software Development Tools

The available tools for addressing the aforementioned challenges of parallel programming can be categorized into four main groups: profiling, debugging, tuning and implementation tools.

3.1 Profiler

Profilers measure the runtime characteristics of a software. They collect information on the functions called during execution, how often they are called and how much time was spent in them. They record the memory usage of the application to show potential leaks or inefficient access patterns. Profilers suited for multicore software development should record these metrics for each thread separately to better reflect the actual runtime characteristics.

With the knowledge gained from profiling the developer can identify the critical parts that will most likely benefit from performance optimization and parallelization. Functions that take long and are called often are obvious candidates. The speedup achieved by parallelizing them can then again be evaluated with the profiler.

3.2 Debugger

Debuggers are essential for analyzing the execution of a software. They allow the developer to reason about the program flow and the detection of logical errors. Although debugging logical errors by statically analyzing the source code is possible most debuggers execute the software and allow the developer to suspend and resume the execution at predefined points and to inspect the data that is present in the software. With non-multi-threaded software the developer only has to follow one trail of execution and can get a good understanding by watching his software execute step-for-step. Multi-threaded code complicates the situation and the debugging tool should offer ways for dealing with the complexity trying to understand the execution and interaction of the different threads. It should be possible to evaluate the coordination aspects of the implementation and to identify parallel errors such as deadlocks and race conditions.

3.3 Tuner

Tuners measure runtime characteristics of software to provide information about the parallel performance. They record slightly different metrics than profilers e.g. they analyze time spent on thread synchronization, load balancing

and scalability. The developer can use the results to further improve the software and to verify his parallel implementation in terms of efficiency and scaling.

3.4 Implementation

The aforementioned tools are analysis tools. They help the developer understand the software and the parallel implementation. Implementation tools help the developer design and write parallel software.

Parallel modeling tools support the developer when exploring the dependencies of the different operations in his software. The developer can see which operations can run concurrently and how they interact with the data structures. When not used for exploring, modeling tools can be used to document the parallel solution.

Parallel libraries and frameworks offer complete implementations of parallel algorithms where the developer fills in the missing code for his use case. There are libraries and programming languages that offer parallel programming abstractions to make it easier for the developer to represent the parallelism found in the design phase. These tools offer specialized profilers, debuggers and tuners to check for inconsistencies in the implementation and warn if they detect errors.

4 Market Study

This section presents the market study. It first defines the most important characteristics for each of the aforementioned tool categories and presents the findings from a survey of tool vendors.

4.1 Participating Tools

The following 18 tools participated in the study:

- Acumem ThreadSpotter, Acumem
- Akka, Scalable Solutions
- Allinea DDT, Allinea
- DataRush, Pervasive Software
- Enterprise Architect, SparxSystems
- GPI, Fraunhofer ITWM
- IESE Simulink Partitioner, Fraunhofer IESE
- Intel Ct, Intel
- JProfiler, ej-technologies
- Offload, Codeplay Software
- PGI, The Portland Group
- Poly-Platform, PolyCore Software
- Prism, CriticalBlue
- PVS-Studio, Program Verification Systems
- SAMG, Fraunhofer SCAI
- Scalasca, Forschungszentrum Jülich
- Visual Studio 2010, Microsoft
- Zoom, RotateRight

4.2 Functionality

The questions on the functionality cover the basic features of each product. These are features common to all products considered in the study.

Product Categorization

Participants were first asked to categorize their product. Depending on the characterization only a subset of the questions was shown later on. Participants that categorized their products as a profiler, debugger or tuner had to fill out general questions about analysis tools first and then questions specific to their tool category. Products which were not categorized as analysis tools were con-

sidered as implementation tools and only the part of the survey specific to implementation tools was shown to the participants.

Table 1 presents the answers of the categorization. The columns labelled »Profiling«, »Debugging« and »Tuning« correspond to the tool categories »Profiler«, »Debugger« and »Tuner« described in chapter 1. The rest of the columns correspond to »Implementation Tools«. They have been split up into seven sub-categories to accommodate the diversity of the category and to better show the distribution of tools in this category.

CASE/Modelling tools are used primarily during the design phase. They are used to develop the structure of a software and form the basis for the specification that is later implemented. Modelling tools for parallel software should include features for identifying opportunities for parallelism and documenting the parallel solution.

Algorithm tools offer an already parallelized solution for a given algorithm. If the algorithm is applicable to the software and the corresponding library can be easily incorporated the developer can improve the performance of his software with minimal effort.

Runtime tools offer additional programming libraries that facilitate the implementation of parallel software by providing higher level abstractions or by providing special implementations of parallel programming primitives.

Framework tools are programming libraries that implement a certain parallel pattern and provide abstractions for implementing a concrete parallel solution. The developer fills out the missing functionality that is specific to his software and lets the framework handle the execution of the parallel pattern.

In the context of multicore software development compilers can provide automatic parallelization of certain algorithmic structures, e.g. loop constructs. If the compiler detects they can be parallelized safely, i.e. without sacrificing correctness, he emits the according instructions automatically during compilation.

Libraries belong to a category similar to algorithms. However, libraries are usually more general and can be used in more scenarios than algorithm tools. Mathematical libraries for example offer special parallelized versions of common operations such as matrix multiplications.

Languages for multicore software development offer programming abstractions for implementing parallel software. These abstractions can hide the resulting parallel execution by completely abstracting the implementation from the execution, e.g. in functional programming languages, or they can make the parallel software more explicit by offering high level constructs to the developer, e.g.

with actor or channel abstractions. Programming languages with high level support for parallel primitives can better check for parallel errors since the parallelism is more explicit.

Vendor	Product	Profiling	Debugging	Tuning	CASE\Modeling	Algorithm	Runtime	Framework	Compiler	Library	Language
Program Verification Systems Co. Ltd	PVS-Studio		●								
Allinea Software	Allinea DDT	●	●	●							
PolyCore Software	Poly-Platform	●		●	●		●	●			
CriticalBlue	Prism	●		●		●		●			
The Portland Group	PGI Suite	●	●	●			●		●		●
ej-technologies GmbH	JProfiler	●	●								
Pervasive Software	DataRush	●	●			●	●	●		●	●
Codeplay Software Ltd	Offload: CE		●				●		●	●	●
Microsoft	Visual Studio	●	●	●	●	●	●	●	●	●	●
Scalable Solutions AB	Akka						●	●	●	●	
SparxSystems Central Europe	Enterprise Architect				●						
RotateRight LLC	Zoom	●		●							
German Research School for Simulation	Scalasca	●									
Acumem	ThreadSpotter	●		●							
Intel	Intel Ct						●	●	●	●	●
Fraunhofer SCAI	SAMG					●		●		●	
Fraunhofer IESE	IESE Simulink partitioner			●	●		●				
Fraunhofer ITWM	GPI						●			●	

Table 1: Product categories

Of the 18 participating tools, 10 were categorized as profiling tools, 7 as debugging tools, 8 as tuning tools and 12 as implementation tools. Several tools can be used for multiple purposes and some vendors participated with a suite of products for multicore software development. The complete category distribution can be seen in Figure 1.

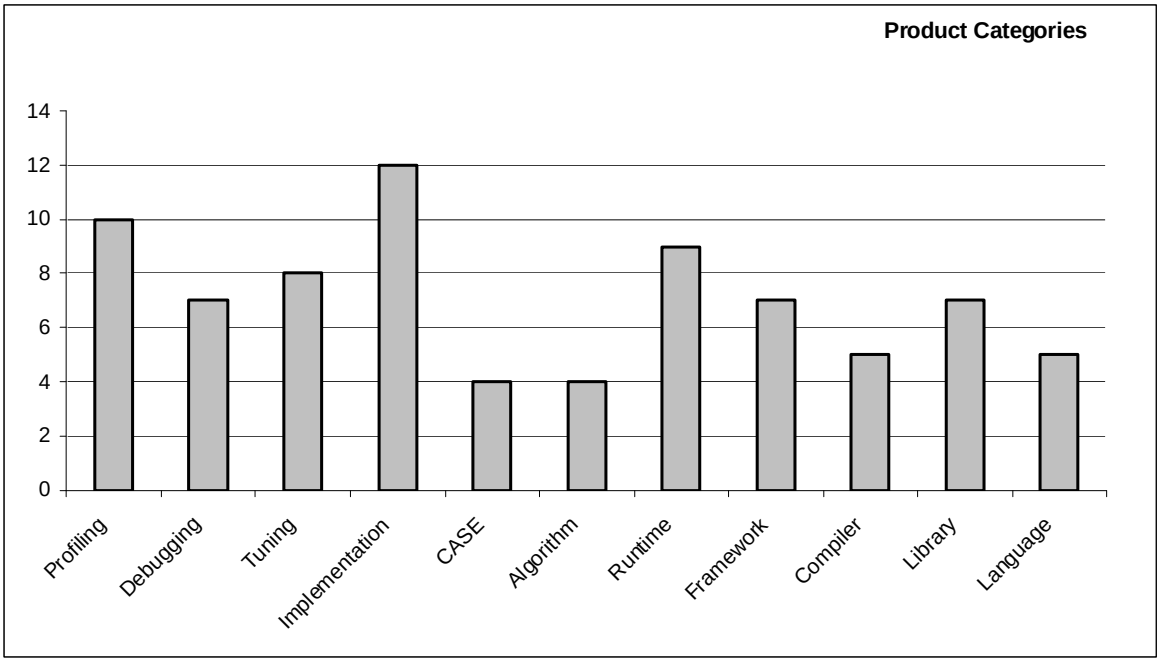


Figure 1: Distribution of product categories

Areas of application

Vendors were also asked to narrow down the area of application of their product. Category and area of application of a product combined provide a first impression of the applicability of the product in a specific scenario. Table 2 shows the answers to this question.

Vendor	Product	Embedded Software	Networking Software	Numerical Simulations	Office Applications	Technical Applications	High Performance Computing	Computer Games
Program Verification Systems Co. Ltd	PVS-Studio			●	●	●	●	●
Allinea Software	Allinea DDT	●	●	●		●	●	●
PolyCore Software	Poly-Platform	●	●	●		●	●	
CriticalBlue	Prism	●	●	●	●	●		●
The Portland Group	PGI Suite			●		●	●	
ej-technologies GmbH	JProfiler		●	●	●	●	●	●
Pervasive Software	DataRush		●	●		●	●	
Codeplay Software Ltd	Offload: CE						●	●
Microsoft	Visual Studio	●	●	●	●	●	●	●
Scalable Solutions AB	Akka		●				●	●
SparxSystems Central Europe	Enterprise Architect	●			●	●		
RotateRight LLC	Zoom	●	●	●	●	●	●	●
German Research School for Simulation	Scalasca			●			●	
Acumem	ThreadSpotter	●	●	●		●	●	●
Intel Corporation	Intel Ct			●		●	●	●
Fraunhofer SCAI	SAMG			●		●	●	
Fraunhofer IESE	IESE Simulink partitioner	●						
Fraunhofer ITWM	GPI			●		●	●	

Table 2: Areas of application

The distribution of the answers can be seen in Figure 2.

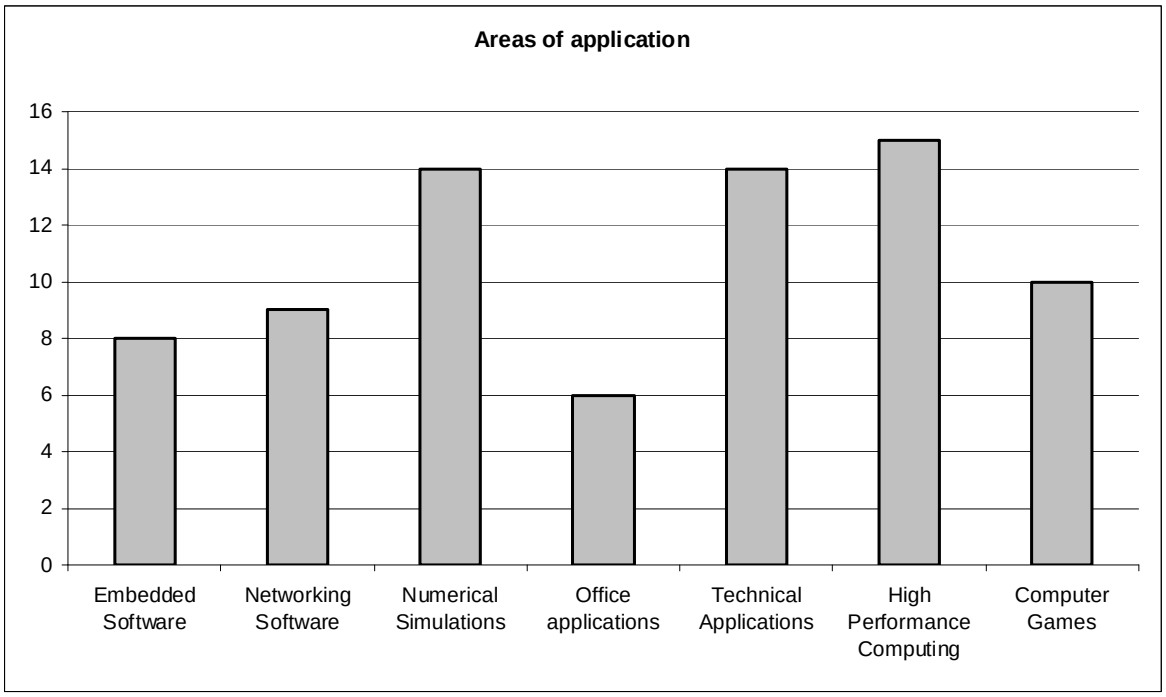


Figure 2: Distribution of answers for area of application

Other basic features are functional requirements like the operating systems and processor architectures the product supports, the interface and the availability of a trial version.

4.3 Tools for analysis

Tools suited for analysis share a common feature set. First to be considered are the analysis requirements. In order to give meaningful results analysis tools are often well adapted for analyzing specific processor architectures or programming languages.

The characteristics of the processor architectures are important for profiling and tuning performance problems. With additional information about cache sizes or bus speeds a tool can check if the program's behavior leads to suboptimal performance on that particular platform. The more knowledge a tool has about a certain characteristic, the more accurate are the results and the interpretation of the results.

Even more important than the supported processor architectures are the programming languages that are supported by an analysis tool. A distinction has

to be made between compiled code that runs natively and code that runs in some kind of virtual machine, either interpreted or compiled. Tools that can analyze native code usually support all languages that are compiled to native code. Tools that analyze code that runs in a virtual machine usually only support that particular virtual machine and programming language. This dependence seems like a disadvantage but is remedied with access to precise execution characteristics that come directly from the virtual machine.

Analysis can be done on different levels. *Model analysis* uses a model of the computation to check for conceptual errors. Models include UML diagrams or library-specific computational graphs. Analyzing on this level reveals errors or misconceptions about the structure of the software. *Static analysis* analyzes the source code without needing to compile and execute the software. It finds subtle errors like missing initializations of variables or unsafe function calls that lead to surprising runtime behavior. *Communication analysis* monitors certain calls of programming libraries during runtime and checks for inefficient communication patterns. Analysis on the *instruction level* is the most common form of analysis and analyzes the individual instructions of the software. It follows the execution of the software and provides the most detailed information about it.

The amount of preparation for an analysis to run is an important characteristic for an analysis tool. The tool and the analysis have to be integrated in the development workflow. The easier this can be done the more often the tool will actually be used. This ensures that developers who are tasked with finding and fixing performance problems can evaluate their changes faster and more precise.

There are different levels of preparation, namely attaching, restarting, automatic instrumentation and manual instrumentation. The easiest way to set up an analysis is if one can simply *attach* the tool to the running software. One can test the software in actual production use and find defects that might be impossible to reproduce in a testing environment. *Restarting* the software in a special analysis environment has the advantage that certain parameters can be set beforehand or special versions of dynamically loaded libraries can be used. This will potentially provide more information about certain behaviors like memory allocation. *Instrumentation* is the process of wrapping source code with special calls to analysis code during compilation. An example is to wrap every function call with a call to an accumulation routine to record the amount of times a function is called during execution or the sequence of called functions. Automatic instrumentation is usually done via a compiler extension and can be globally applied (wrapping each and every function call for example) or specific to a library (wrapping only calls to the library). Manual instrumentation is done by the developer and can only reasonably be applied to small parts of the source code. The developer has the chance to decrease the amount of in-

formation during analysis and focusing on the critical paths of the software. The challenge is to find out which are the critical paths.

After performing the analysis the user must interpret the results. The analysis tool can help him in different ways. First, it has to present the results in an adequate way. This can be done in a textual summary, possibly sorting the analyzed metrics by importance. It can be done by providing a graphical overview to let the user get a picture of the analysis before concentrating on the important parts. An important aspect of the analysis for the user is if it helps him understand where and why his code behaves in the way it does. Most important for him is the mapping from a result to the source code lines that caused it. An analysis tool should give him the information he needs to find and fix the source code.

After modifying the source code the developer has to verify the improvements. He analyzes the software again and compares the new results with the old results. An analysis tool should make this as easy as possible, possibly showing a side-by-side comparison or highlighting the improvements from the previous run. If the profiler stores the results of completed analyses the improvement can be documented over multiple iterations.

Analyzing the runtime behavior of a software always induces some overhead. In general the amount of overhead should be kept as low as possible. This is especially important for behavior that is sensitive to changes in timing, e.g. communication over a network or scheduling in multicore software.

When analyzing multi-threaded software it is important for the developer to have a good overview of the status of the threads. The number of threads running in future applications is going to rise with the number of available cores and analysis tools have to analyze and display results in a way that lets the developer identify potential issues easily. An analysis tool should pose no limits on the amount of threads it is able to analyze and provide useful visualization for. This has to be carefully weighted against the generated overhead when analyzing many threads at once.

Quick-View Analysis Part 1

Product	**PVS-Studio**	**DDT**	**Poly-Platform**	**Prism**	**PGI Suite**	**JProfiler**	**DataRush**
Vendor Company	Program Verification Systems Co. Ltd	Allinea	PolyCore Software	CriticalBlue	The Portland Group	ej-Technologies	Pervasive Software
Processor support	x86, x86_64	x86, x86_64, ia64, Sparc, PowerPC, CellBE, NVidia GPUs	x86, x86_64, ARM, PowerPC, BF561, C55	x86, x86_64, ARM, PowerPC, MIPS32, Renesses SH4, Cavium cn, MIPS, NEC V850	x86, x86_64, NVidia CUDA-enabled devices	x86, x86_64, ia64, Sparc, PowerPC	x86, x86_64, ia64, Sparc, PowerPC, any Java 6 JVM
Language support	C, C++	C, C++, Fortran, .NET, CUDA	C	C, C++	C, C++, Fortran	Java	Pervasive DataRush Dataflow Language Analysis
Analysis							
Level	Static analysis	Static analysis, Communication, Instruction	Instruction	Static analysis, Communication, Instruction	Instruction	Instruction	Model, Static analysis, Communication
Preparation							
Restart before analysis	N/A	○	N/A	○	○	●	●
Compilation before analysis	N/A	●	●	○	○	○	○
Manual modification before analysis	N/A	●	○	○	○	○	○
Results							
Results are stored	●	●	●	●	●	●	○

Product	PVS-Studio	DDT	Poly-Platform	Prism	PGI Suite	JProfiler	DataRush
Vendor Company	Program Verification Systems Co. Ltd	Allinea	PolyCore Software	CriticalBlue	The Portland Group	ej-Technologies	Pervasive Software
Presentation		Graphical	Graphical	Graphical	Textual, Graphical	Graphical	Graphical
Availability		During, After processing	After processing	After stopping	After processing	During, After stopping	During
Mapping	N/A	N/A	N/A	●	N/A	●	●
Overhead	N/A	2-3%	N/A	Depends on method used	2%	N/A	Low
Limit	N/A	No	N/A	No	256 Processes, 64 Threads per process	No	No

Table 3: Quickview Analysis Tools Part 1

Quick-View Analysis Part 2

Product	**Offload: Community Edition**	**Visual Studio 2010**	**Zoom**	**Scalasca**	**Thread-Spotter**	**IESE Simulink partitioner**
Vendor Company	Codeplay	Microsoft	RotateRight	Forschungszentrum Jülich	Acumem	Fraunhofer IESE
Processor support	x86	x86, x86_64, ia64, ARM	x86, x86_64, ARM, PowerPC, Cell BE	x86, x86_64, ia64, Sparc, PowerPC, MIPS, NEC	x86, x86_64	x86, x86_64, ARM, FPGA-based processors
Language support	C++	C++, C#, .NET, Python, DirectX Compute	C, C++, Fortran	C, C++, Fortran	C, C++, Fortran, any compiled language	Simulink / generic data-flow based languages
Analysis						
Level	Static analysis, Instruction	Model, Static analysis, Communication, Instruction	Static analysis, Instruction	Communication, Function/routine, Load balance	Instruction, Memory access	Model, Static analysis
Preparation						
Restart before analysis	●	○	○	●	○	○
Compilation before analysis	○	○	○	●	○	○
Manual modification before analysis	○	○	○	●	○	○
Results						
Results are stored	N/A	●	●	●	●	○
Presentation	Textual	Textual, Graphical	Textual, Graphical	Textual, Graphical	Textual, Graphical, HTML Output	Textual
Availability	After stopping	After stopping	After processing	After processing	After processing	After processing

Product	**Offload: Community Edition**	**Visual Studio 2010**	**Zoom**	**Scalasca**	**Thread-Spotter**	**IESE Simulink partitioner**
Vendor Company	Codeplay	Microsoft	RotateRight	Forschungszentrum Jülich	Acumem	Fraunhofer IESE
Mapping	N/A	●	●	●	●	N/A
Overhead	N/A	Depends on desired analysis details	2%	Depends on configuration / application	Depends on configuration, 10% for long running applications	None, analysis runs at compile time
Limit	○	N/A	○	● Depends on available processors and memory	○	N/A

Table 4: Quickview Analysis Tools Part 2

4.3.1 Profilers

Profilers belong to the analysis tools category. The following section presents additional characteristics of profiling tools identified for the market study. The following paragraphs describe each characteristic in detail.

Type of profiling

System-wide analysis examines the execution of the whole system which includes the developer's software as well as all the other running programs and the operating system. This kind of analysis is most useful for evaluating a software in the target environment and to understand how it interacts with the system and its resources. Since the whole system is analyzed, the amount of gathered data can be huge and has to be adequately presented. *Software-specific analysis* is the most common form of profiling. The profiler analyzes only the behavior of the software in question and provides a detailed picture of the behavior. *Library-specific* profilers specialize on certain libraries and analyze the usage of the library in question. The behavior can usually be modified to allow for high-level questions such as efficient use of certain library calls.

Method of profiling

Closely related to the kind of analysis is the method of profiling. *Simulation* starts the software in a virtual environment and simulates the execution while recording every instruction as it is executed. This method is very accurate but generally generates a lot of overhead during analysis. *Statistical sampling* records the software's state in defined intervals and accumulates the running times of functions. The software usually does not have to be modified in order to be sampled. However, re-compiling and generating debug symbols improve the interpretation of the result later on. *Instrumentation* requires modification of the software. This modification can be done manually by adding profiling calls to the source code or automatically during compile time by automatically wrapping function calls with profiling calls. The advantage is that instrumentation can be selectively applied so that only the most relevant information will be extracted. This requires that the most relevant places for instrumentation have to be identified first. *Event-based* profiling uses special monitoring hardware on the processor to record certain hardware events which relate to performance issues, such as pipeline stalls or poor memory access patterns.

Library Support

Support for certain libraries means that the tool includes special profiling instruments for analyzing the usage of the respective libraries. It can therefore facilitate the analysis of the results by showing only those relevant to the use of

the library. This makes it easier for the developer to see performance deficits related to his understanding of the library.

Profiling information

The profiler collects data about the execution of the software. In addition to the accumulation of function calls profilers might record more context information. Without context information the developer only sees how often a function was called, but not from where. By recording the call graph – the hierarchy of function invocations at the moment of the function call – the profiler can provide more details and the developer can see which code paths lead to critical behavior. If the profiler also records a trace of the software the developer can essentially play back the execution of his software. This gives him additional power in finding critical parts of his software.

Profiling multicore software

For the analysis of multi-threaded software there are additional characteristics that influence software performance. These include the CPU load caused by the individual threads and the memory-, I/O- and cache access. The CPU load is an indicator for how well the software scales to more processors. If not all the cores are used during an extensive operation then there is potential for more parallelism. Memory-, I/O- and cache are hardware resources that are shared by all threads so that access has to be coordinated. By profiling the access patterns the developer can see which resources are contested and if the coordination leads to performance problems. Thread coordination often leads to waiting times in the software, e.g. waiting for access to I/O devices, waiting for data parallel threads to finish or waiting for acquiring a lock around a shared variable. Analyzing and improving the cause of these waiting times is a very fine-grained optimization step.

Quick-View Profiler Part 1

Product	**DDT**	**Poly-Platform**	**Prism**	**PGI Suite**	**JProfiler**
Vendor Company	Allinea	PolyCore Software	CriticalBlue	The Portland Group	ej-Technologies
Profiling					
Type	Software-specific, Library-specific	Software-specific	Software-specific	System-wide, Software-specific, Library-specific	Software-specific
Supported Libraries	MPI, CUDA, PAPI	N/A	N/A	MPI, OpenMP, PGI Accelerator	N/A
Method	Statistical sampling, Instrumentation, Event-based	Instrumentation	Instrumentation, Simulation	Statistical sampling, Instrumentation, Event-based	Statistical sampling, Instrumentation, Event-based
Gathered Information	Flat profile, Callgraph, Trace, HW Performance Counters	Flat profile, Callgraph	Callgraph with Trace	Flat profile, HW Performance Counters	Callgraph, Trace
Multicore profiling					
CPU load	●	○	●	●	●
Memory	○	○	●	●	●
I/O	○	○	○	○	●
Cache	●	○	●	●	○
Further					Extensive analysis of monitor contentions and wait states
Multicore idle times					
I/O	●	○	○	○	●
Synchronization	●	○	●	○	●
Acquiring Locks	●	○	●	○	●

Table 5: Quickview Profiler Part 1

MWARE multicore software

Quick-View Profiler Part 2

Product	**DataRush**	**Visual Studio 2010**	**Zoom**	**Scalasca**	**ThreadSpotter**
Vendor Company	Pervasive Software	Microsoft	RotateRight LLC	German Research School for Simulation Sciences GmbH	Acumem
Profiling					
Type	Library-specific	System-wide, Software-specific, Library-specific	System-wide, Software-specific	Software-specific, Library-specific	Software-specific
Supported Libraries	DataRush and other Java libraries	MPI, OpenMP, Intel Threading Building Blocks	N/A	MPI, OpenMP	N/A
Method	Statistical sampling	Statistical sampling, Instrumentation, Event-based	Statistical sampling, Event-based	Instrumentation, Event-based	Sampling of memory accesses
Gathered Information	Callgraph	Callgraph	Callgraph, Trace, HW Performance Counters	Callgraph, Trace, HW Performance Counters	Callgraph
Multicore Profiling					
CPU load	●	●	●	●	○
Memory	●	●	○	○	●
I/O	●	●	○	●	○
Cache	○	○	○	●	●
Further					Classificaton of slowspots, Various forms of Locality, Poor memory access patterns, Cache contention, Bandwidth limitations, Prefetcher effectiveness, Thread communication patterns,

Product	DataRush	Visual Studio 2010	Zoom	Scalasca	ThreadSpotter
Vendor Company	Pervasive Software	Microsoft	RotateRight LLC	German Research School for Simulation Sciences GmbH	Acumem
					False sharing
Multicore Idle times					
I/O	●	●	○	○	○
Synchronization	●	●	○	●	○
Acquiring Locks	○	○	○	○	○
Further				Mostly waiting times related to MPI such as waiting for messages & waiting in synchronizing operations	

Table 6: Quickview Profiler Part 2

4.3.2 Debuggers

Debuggers belong to the analysis tools category. The following section presents additional characteristics of debugging tools.

Type of debugging

Developers use debuggers to examine the execution of their software and to detect conceptual errors rather than syntactical errors or performance issues. Debuggers usually work by executing the software in their own environment and stopping the execution when errors occur, e.g. on null pointer exceptions, or when the execution arrives at predefined points, called breakpoints. The developer can then examine the state of the software, step through the execution instruction by instruction and figure out the source of the problem.

In addition to this runtime analysis some debuggers analyze the source code without running the software. This kind of static analysis is most commonly used to check for security errors that share a common pattern, like buffer overflows. When analyzing multicore software static analysis can reveal parallelization errors like unsynchronized access of a shared variable which may be hard to find using normal debugging.

Debugging multicore software

When debugging multi-threaded software it is important for the developer to know when common parallelization errors occur. A debugger should identify the most common ones like deadlocks, livelocks, race conditions and memory leaks. Further, the debugger should aid the developer by providing a mapping from the error to the causing source code. This is a difficult challenge because the problem might lie in two different parts of the source code and has to be adequately presented to the developer.

Trace analysis

The recording of a program's execution is called a trace. It is useful for debugging purposes since it records the sequence of function calls and their parameters during execution. This way, the execution of a software can later be dissected to determine the cause of a bug. The manufacturers were asked if their debuggers collect and analyze existing trace data and in which format they store it.

Quick-View Debuggers Part 1

Product	**PVS-Studio**	**DDT**	**PGI Suite**	**JProfiler**	**DataRush**	**Offload: CE**	**Visual Studio 2010**
Vendor Company	Program Verification Systems Co. Ltd	Allinea	The Portland Group	ej-Technologies	Pervasive	Codeplay Software Ltd	Microsoft
Debugging							
Type		Static analysis	Runtime analysis with support for MPI, OpenMP and PGI Accelerator model	Runtime analysis	Runtime analysis with support for DataRush and other Java libraries		Static analysis, Runtime analysis
Multicore Debugging							
Deadlocks	○	●	○	●	●	○	●
Livelocks	○	●	○	●	●	○	●
Race Conditions	○	○	○	○	●	○	●
Memory leaks	○	●	○	●	●	○	○
Mapping to source code		For all errors		For all errors			For deadlocks
Usage of HW Peformance Counters	N/A	●	●	○	○	N/A	●
Debug Traces							
Traces are created	N/A	●	○	●	●	N/A	●
Trace format	N/A	SQL database	○	Proprietory	Visual VM Application Graphs	N/A	Proprietory

Product	PVS-Studio	DDT	PGI Suite	JProfiler	DataRush	Offload: CE	Visual Studio 2010
Vendor Company	Program Verification Systems Co. Ltd	Allinea	The Portland Group	ej-Technologies	Pervasive	Codeplay Software Ltd	Microsoft
Analysis of existing traces	N/A	○	○	○	●	N/A	●

Table 7: Quickview Debuggers

4.3.3 Tuners

The different threads of a multi-threaded software will be run in parallel on multicore processors. Tuning is used to assess and further improve the performance. Tuners use profiling to measure the important metrics for multicore performance.

Type of tuning

Tuners usually use profiling techniques to gather performance information. *Statistical sampling* records the software's state in defined intervals and accumulates the running times of functions. *Instrumentation* requires modification of the software by adding profiling calls to the source code or automatically during compile time. Instrumentation can be selectively applied so that only the most relevant information will be extracted. *Event-based* profiling uses special monitoring hardware on the processor to record certain hardware events which relate to performance issues, such as pipeline stalls or inefficient memory access patterns.

Analyzed performance characteristics

The metric *processor load* shows how the software distributes the workload on the available resources and if it will scale to higher numbers of processor cores in the future. The metrics *memory access*, *cache access* and *false sharing* are related to accessing data from memory and how long the threads have to wait. Since a thread is blocked while he waits for data to arrive, another thread can do work with the data he has. Tuners show if such overlapping occurs in the software.

False sharing occurs when two threads are scheduled on different processor cores but access data in a memory region that maps to the same cache line. Each modification of the data in the cache by one of the threads results in the cache coherency module of the processor trying to keep the caches of both cores in synch. This will happen even if the two threads operate on different, non-overlapping data segments. Cache synchronization leads to memory stall time while waiting for the synchronization to happen. This leads to performance degradation and has to be handled.

Further performance degradation can be caused by inefficient thread synchronization. This might be caused by over-subscribing a shared resource so that the time spent waiting for acquiring a lock outweighs the time spent for using the resource. Another cause might be an imbalanced distribution of work among threads so that some threads have to wait for others to finish at the end of a distributed computation.

Additional features

The usage of hardware performance counters gathers data that is very hardware-dependent and low-level. By using this data a tuning tool can provide very detailed information about the processor's functional, cache and memory units. This enables the developer to further tune his software to the specific processor architecture.

Tuning tools that support special libraries provide special profiling instruments for in-depth analysis of their usage in the software.

Quick-View Tuners

Product	**DDT**	**Poly-Platform**	**Prism**	**PGI Suite**	**Visual Studio 2010**	**Zoom**	**Thread-Spotter**	**IESE Simulink Partitioner**
Vendor Company	Allinea	PolyCore Software	CriticalBlue	The Portland Group	Microsoft	Rotate-Right LLC	Acumem	Fraunhofer IESE
Tuning								
Method	Statistical sampling, Instrumentation,	Instrumentation	Instrumentation	Statistical sampling, Instrumentation, Event-based	Statistical sampling, Instrumentation, Event-based	Statistical sampling, Event-based	Statistical sampling, Instrumentation	Model checking, semantic analysis, flow analysis
Characteristics								
Processor/Core load	●	○	●	●	●	●	○	●
Memory access	○	○	●	●	●	●	●	●
Cache access	●	○	●	●	○	●	●	●
Thread communication	○	○	●	●	●	○	●	●
False sharing	○	○	●	○	○	○	●	○
Misc.								
HW Performance Counters	●	○	N/A	●	●	●	○	○
Library support	MPI	○	○	MPI, OpenMP, PGI Accelerator model	MPI, OpenMP, Intel Threading Building Blocks	○	○	○

Table 8: Quickview Tuners

4.4 Tools for Implementation

The following section presents the questions on tools suited for implementing parallel software.

Fundamental concepts

The implementation of parallel software is guided by the application of certain fundamental principles/abstractions. While making a software multi-threaded is almost trivial under today's operating system and their frameworks, developing software that maintains correctness and scales efficiently with hardware changes requires a more conscious approach to parallelization.

Research on parallel and distributed computing has developed several concepts for dealing with parallelism in software. These concepts differ in their level of abstraction, their handling, their fault tolerance and their performance. The concepts don't necessarily stand on their own and can be combined. The following programming concepts were identified for this study:

- Threads
- Message passing
- Dataflow concepts
- Functional programming concepts
- Asynchronous computation with callbacks
- Lock-free data structures
- Parallel patterns

The most used concept for developing parallel software is splitting the computation into multiple threads. This can be done by hand using operating system primitives or using thread libraries like pthreads or boost::threads. It can also be done automatically by annotating the source code with certain compiler directives that will get transformed into the right calls to the threading library at compile time. Programming with threads requires special knowledge of the advantages and disadvantages. It is regarded as a fairly low level technique. The developer has to be aware of a lot of issues that don't normally arise in sequential programming. The multiple threads of execution are interleaved nondeterministically by the operating system's scheduler. Special care has to be taken that critical data can only be written to by one thread and that resources are acquired in such a way that the system doesn't lock up. Programming with threads directly is said to be the assembler language of parallel computing.

Lock-free data structures are designed to be safe and efficient in the context of multi-threaded access. They usually depend on functions that are guaranteed to be executed atomically, i.e. non-interruptible by other threads, by the proc-

essor. Building on these functions, lockless data structures can provide consistent access without forcing threads to acquire a lock first. Developers use these data structures in places where they share data explicitly between threads and want to make sure that it stays consistent.

Asynchronous computation with callbacks is a higher level concept than using threads explicitly. The computation is done in a background thread so that the main thread can continue its work. On completing the computation, a callback function is invoked with the result. This concept is commonly used in graphical applications where an expensive computation cannot block the main thread responsible for updating the user interface.

Message passing is a fundamental concept that places restrictions on how data is accessed in a software system. By default no data is shared and computation is done via sending messages containing data to be processed. The result is then sent back afterwards. This concept provides an abstraction that makes it easier to reason about data access in parallel and distributed systems and prohibits race conditions by design.

Dataflow concepts provide a model that looks at the software from the perspective of data dependencies. Operations in the software depend on their input data and produce output data. The developer models the software by connecting the operations' inputs with other operations' outputs. The resulting model can be analyzed to identify operations that can run in parallel and operations that must run sequentially. This information can then be used to automatically distribute and schedule the operations on the available cores.

Functional programming adheres to the mathematical definition of a function, in that a function has no side effects and no internal state so that the output depends solely on the input. Software written in a functional style is mostly composed out of functions, the data itself is immutable. These characteristics are ideal for parallel processing since the application of side-effect free functions can be easily and safely distributed on parallel systems.

Parallel patterns are established algorithmic structures to be reused in concrete use cases. For a given problem they describe the parallelization strategy that a developer can adopt and complete for his application. Tools usually provide parallel patterns in a framework and a developer provides the concrete operations, the parallel execution is subsequently managed by the tool's runtime.

Abstractions

Implementation tools offer certain abstractions for the developer to express the parallel operation in the software. These abstractions differ in the way they help the developer with reasoning about creation of threads, coordination of

threads and overall performance. The more high level the abstraction the more can be automatically inferred from the implementation. However, there are also more implications to consider.

Threading libraries offer a fast way to create threads and coordinate them afterwards. The developer has more control over the implementation but also more responsibility to check for errors in his code. Parallel libraries and parallel pattern frameworks offer complete parallel implementations. The developer has to follow the respective programming interface guidelines. These determine how data has to be prepared and accessed during computation. Actors are an abstraction similar to a thread in that they encapsulate a unit of work. Their internal state is hidden and can only be queried or changed by sending messages. Due to the encapsulation actors provide safety from race conditions. Actors offer a model for asynchronous and concurrent computation and are usually scheduled automatically. However the developer has to structure his software accordingly to use actors effectively.

Error Prevention

An important feature of an implementation tool is the ability to prevent common parallelization errors either by making it impossible to construct them using their abstraction or to detect it during compilation or execution. Some of the above concepts lend themselves better to this than others. If it is not possible from within the tool itself, the developer should have a chance to review the correctness of his solution by debugging it using either standard or special tools suited for use with the implementation tool.

Programming language

An implementation tool always affects the choice of programming language. Although high level abstractions like parallel patterns can be expressed in many programming languages, a concrete tool that offers them in a framework usually supports only a subset. The same goes for libraries that offer parallel algorithms. They have to be chosen according to the software's target programming language in order to be used.

Support

In order to use an implementation tool effectively, it must be easy for the developer to get started with it. The developer has to understand the abstractions, how to use them effectively and how to diagnose errors. The tool should offer the required documentation for getting started, advanced usage examples and in-depth analysis. Ideally, the manufacturer can provide support ranging from personal on-site support to an active community exchanging ideas on how to use the product.

Quick-View Implementation Part 1

Product	**Poly-Platform**	**Prism**	**PGI Suite**	**DataRush**	**Offload: CE**	**Visual Studio 2010**
Vendor Company	PolyCore Software	CriticalBlue	The Portland Group	Pervasive Software	Codeplay Software Ltd	Microsoft
Concept	Threads, Message passing, Dataflow, Functional, Asynchronous	Threads, Dataflow, Functional, Lock-free data structures, Parallel patterns	Threads, Message passing, Parallel patterns	Dataflow, Functional, Asynchronous, Lock-free data structures, Parallel patterns	Dataflow, Functional	Threads, Functional, Asynchronous, Lock-free data structures, Parallel patterns
Memory model	Shared memory, Distributed memory, NUMA	Shared memory	Shared memory, Distributed memory, NUMA	Shared memory, Distributed memory, NUMA	Shared memory, Distributed memory, NUMA	Shared memory
Programmer knowledge	Functional programming		Thread programming	Dataflow programming, Parallel patterns	Functional programming	Thread programming
Programming abstraction	Library calls, Actors	Threads	Threads, Actors	Parallel patterns	Library calls	Threads, Tasks
Validation of correctness with standard tools	●	○	●	●	●	●
Special tool support	Poly-Mapper validates the communications Topology	Prism has a set of verification features to help the programmer ensure that the parallel code is safe.	PGDBG parallel debugger	VisualVM plug-in		Parallel Debugger
Programming language	C, C++	C, C++	C, C++, Fortran	Java	C, C++	C, C++, C#, VB.NET, F#, Python, Ruby
Error prevention						

Product	Poly-Platform	Prism	PGI Suite	DataRush	Offload: CE	Visual Studio 2010
Vendor Company	PolyCore Software	CriticalBlue	The Portland Group	Pervasive Software	Codeplay Software Ltd	Microsoft
Deadlock detection	○	○	○	●	○	○
Race condition detection	○	●	○	●	○	○
Transactional memory	○	○	○	○	○	●
Lock-free data structures	○	○	○	●	○	○
Functional semantics	○	○	○	○	○	○
Dataflow semenatics	○	○	○	●	○	○
Further	Thread safe message passing		Language compliance, Runtime error checking			
Help						
Getting started	Familiar programming language, Familiar abstractions, Plugins for IDEs, Specialized IDEs	Familiar programming language, Familiar abstractions, Plugins for IDEs, Specialized IDEs	Familiar programming language, Familiar abstractions, Parallel Infrastructure, Plugins for IDEs	Familiar programming language, Familiar abstractions, Plugins for IDEs	Familiar programming language, Parallel infrasctructure, Plugins for IDEs	Familiar programming language, Familiar abstractions, Specialized IDEs
Tool support						
Documentation						
Meaningful API documentation	●	●	●	●	●	●
Beginner tutorials	●	●	●	●	●	●

Product	Poly-Platform	Prism	PGI Suite	DataRush	Offload: CE	Visual Studio 2010
Vendor Company	PolyCore Software	CriticalBlue	The Portland Group	Pervasive Software	Codeplay Software Ltd	Microsoft
Usage examples	●	●	●	●	●	●
Further	Reference Manuals		Language reference, Compiler and Tools manuals			
Support						
On-site support	●	●	●	●	●	●
Phone support	●	●	●	●	○	●
Email support	●	●	●	●	●	●
Active community (website, wiki, mailing list, newsgroup, etc.)	●	●	●	●	●	●
Conferences	●	●	●	●	○	●
Seminars/Workshops	●	●	●	●	○	●
Other/Further			Webinars, Tutorials			Startup programs for partners

Table 9: Quickview Implementation Tools Part 1

MWARE multicore software

Quick-View Implementation Part 2

Product	**Akka**	**Enterprise Architect**	**Intel Ct**	**SAMG**	**Simulink partitioner**	**GPI**
Vendor Company	Scalable Solutions AB	SparxSystems Central Europe	Intel Corporation	Fraunhofer SCAI	Fraunhofer IESE	Fraunhofer ITWM
Concept	Threads, Message passing, Dataflow, Functional, Asynchronous, Lock-free data structures, Parallel patterns			Threads	Threads, Message passing, Dataflow, Functional, Asynchronous, Lock-free data structures, Parallel patterns	Lock-free data structures, Parallel patterns, One sided communication, Asynchronous communication
Memory model	Shared memory, Distributed memory		Shared memory, Distributed memory, NUMA	Shared memory, Distributed memory, NUMA	Shared memory, Distributed memory, NUMA	Shared memory, Distributed memory, NUMA
Programmer knowledge			Functional, Parallel patterns, Metaprogramming concepts	Numerical modeling	Dataflow programming, Platform knowledge	Thread programming, Parallel patterns, Understanding of how to achieve maximal locality
Programming abstraction	Actors		Computation kernels, Parallel patterns	Library calls, Actors	Parallelism is derived from data-flow specification	Threads, Communication queues (channels)
Validation of correctness with standard tools	●	○	●	●	●	●
Special tool support			Data visualization tools are also provided			
Programming language	Java, Scala		Is programming language, C, C++ through virtual machine API	C, C++, Fortran	Simulink, ASCET	C, C++, Fortran

Product	**Akka**	**Enterprise Architect**	**Intel Ct**	**SAMG**	**Simulink partitioner**	**GPI**
Vendor Company	Scalable Solutions AB	SparxSystems Central Europe	Intel Corporation	Fraunhofer SCAI	Fraunhofer IESE	Fraunhofer ITWM
Error prevention						
Deadlock detection	○	○	○	○	○	○
Race condition detection	○	○	○	○	●	○
Transactional memory	●	○	○	○	○	○
Lock-free data structures	○	○	○	○	●	○
Functional semantics	●	○	●	○	○	○
Dataflow semenatics	●	○	○	○	●	○
Further			Since race conditions cannot be written, they do not have to be debugged. Determininistic execution is guaranteed.			
Help						
Getting started	Familiar programming language, Familiar abstractions		Familiar programming language, Familiar abstractions, Parallel infrastructure, Plugins for IDEs	Simple library interface, Demo programs	Specialized IDE	Familiar programming language, Familiar abstractions, Parallel infrastructure
Tool support	Standard Java profilers such		Intel(R) Thread Profiler, Intel(R)	MPI tracing tools (e.g.	Simulink Real-time Workshop	gdb, vampir, ddt, valgrind

Product	Akka	Enterprise Architect	Intel Ct	SAMG	Simulink partitioner	GPI
Vendor Company	Scalable Solutions AB	SparxSystems Central Europe	Intel Corporation	Fraunhofer SCAI	Fraunhofer IESE	Fraunhofer ITWM
	as YourKit, JProfiler etc.		VTune(TM) Performance Analyzer	Vampir, ITAC), Threading tools		
Documentation						
Meaningful API documentation	●	○	●	○	○	●
Beginner tutorials	●	○	●	●	○	●
Usage examples	●	○	●	●	○	●
Further	Complete reference				Internal reference documentation/reports	
Support						
On-site support	●	○	●	●	○	○
Phone support	●	○	●	●	○	○
Email support	●	○	●	●	○	●
Active community (website, wiki, mailing list, newsgroup, etc.)	●	○	●	○	○	○
Conferences	●	○	●	○	○	○
Seminars/Workshops	●	○	●	●	○	○
Other/Further			Webinars, Tutorials		Product is intended for in-house use.	

Table 10: Quickview Implementation Tools Part 2

5 Tools

5.1 Acumem ThreadSpotter, Acumem

Company Information

Vendor	Acumem
Established in	2006

Headquarters	Kungsgatan 16 75332 Uppsala Sweden
Telephone	+46 18 130700
Fax	
Website	www.acumem.com
Summary	Acumem makes advanced optimization tools for programmers, that focus on productivity and automatic analysis.

German Service Partner	No

General Product Information

Product name	Acumem ThreadSpotter
Market entry	2008
Version	Acumem ThreadSpotter 2010.1
Description	Acumem ThreadSpotter is an optimization tool for programmers. It diagnoses performance problems with respect to memory bandwidth, cache usage, thread interaction and communication. It presents advice to the programmer along with detailed instructions on how to change the code.

Product categories	☒ Profiling	☐ Debugging
	☒ Tuning	☐ CASE / Software Modeling
	☐ Algorithm	☐ Runtime
	☐ Framework	☐ Compiler
	☐ Library	☐ Programming language
	☐ Other	
Product markets	☒ Embedded software	☒ Networking software
	☒ Numerical simulations	☐ Office applications
	☒ Technical applications	☒ High Performance Computing
	☒ Computer Games	☒ Other: Any memory bound, cpu bound or multithreaded application.

Customer references	ORNL, CEA, NERSC, TACC

Software Development

Software development process model	☒ All	
	☐ V model	☐ Waterfall
	☐ Agile:	☐ Other
Recommended stages	Optimization is traditionally handled in the final stages of a project. Premature optimization can sometimes introduce hard-to-find errors and render the code more difficult to maintain. Despite this, we recommend using this tool throughout the development process to find problem spots early on, and to educate the programmers on lean programming styles.	
Other related products		
Related products in combination		

Functionality

Supported installation architectures	☒ x86	☒ x86_64
	☐ ia64	☐ ARM
	☐ Sparc	☐ PowerPC
	☐ Cell Broadband Engine	☐ GPUs:
	☐ Other	

Supported operating systems	☒ Windows: XP, Vista, 7	
	☒ Linux: Debian, Fedora, Mandriva, RedHat, SUSE	
	☐ AIX	☐ HP-UX
	☐ BSD: FreeBSD	☒ Solaris
	☐ Mac OS X	☐ Other

Application type	☒ Graphical application	☒ Commandline application
	☐ Plugin: Visual Studio	☐ Library/Framework
	☐ Other	

Trial version available	Yes
Time limit	14 days
Limitations	The trial is for the companion product, Acumem SlowSpotter, which does not include multithread related advice.

Analysis

Supported analysis architectures	☒ All the architectures the product runs on	
	☒ x86	☒ x86_64
	☒ ARM	☒ Sparc
	☒ PowerPC	☐ Cell Broadband Engine
	☐ GPUs	☒ Other: Any processor can be described and fed into the analysis model.
Supported languages	☒ C	☒ C++
	☒ Fortran	☐ Java
	☐ C#	☐ Python
	☐ .NET	☐ OpenCL
	☐ DirectX Compute	☐ ATI Stream
	☒ Other: Any compiled language	

Analysis level	☐ On the model level (e.g. UML verification, process checking, sequence diagram analysis)
	☐ On the source code level (e.g. static analysis)
	☐ On the communication level (e.g. between processes/threads in MPI)
	☒ On the instruction level (e.g. runtime analysis, profiling)
	☒ Other/Further levels: Analysis is performed on a memory access level.

Analysis preparation	☒ No preparation necessary ☐ Restart in analysis environment ☐ Compilation with special parameters: ☐ Manual modifications: ☐ Adding compiler directives ☐ Adding library calls ☐ Other

Analysis results are stored	Yes

Presentation of analysis results	☒ Textual summary ☒ Graphical presentation ☒ Other: Set of HTML pages

Required practice before using the tool effectively	☐ ☐ ☒ ☐ ☐ Very little A lot
Special knowledge required for analysis	No

Analysis results mapping to code	☒ Yes
Automated advice	☒ Yes
Availability of results	☐ During the analysis ☐ After stopping the analysis ☒ After stopping the analysis and preparing the results
Comparison of analysis runs	Yes
Processing of analysis results is possible with other tools	No

Analysis overhead	Depends on settings. Typically 10% for long running applications.
Analysis limit	No
Presentation limit	No

MWARE multicore software

Profiling

Profiling type	☐ System-wide analysis (like DTrace or oprofile) ☒ Software-specific analysis (like gprof) ☐ Library-specific analysis (like TAU for MPI) ☐ MPI ☐ OpenMP ☐ Intel Threading Building Blocks ☐ OpenCL ☐ DirectX Compute ☐ CUDA ☐ ATI Stream ☐ Other ☐ Other
Profiling method	☐ Statistical sampling ☐ Instrumentation ☐ Simulation ☐ Event-based ☒ Other: A unique kind of sampling of memory accesses.
Gathered profiling information	☐ Combined runtimes (flat profile) ☒ Runtimes with callgraph ☐ Runtimes with callgraph and trace

Usage of hardware performance counters	No
Features for multi-core analysis	☐ CPU load per thread/process ☒ Memory profiling ☐ I/O profiling ☒ Cache profiling ☒ Other: Classificaton of slowspots, Various forms of Locality, Poor memory access patterns, Cache contention, Bandwidth limitations, Prefetcher effectiveness, Thread communication patterns, False sharing
Analysis of processor idle time	No
Measured idle times	☐ Waiting for I/O ☐ Waiting for thread synchronization ☐ Waiting for acquiring locks ☐ Other

Tuning

What kinds of characteristics does your product analyze for performance evaluation?	☐ Processor/Core load	☒ Memory access
	☒ Cache access	☒ Thread communication
	☒ False Sharing	☐ Other

What kinds of techniques does your product use for analysis?	☒ Statistical sampling ☐ Simulation ☐ Other	☒ Instrumentation ☐ Event-based analysis
Does your product use hardware performance counters during analysis?	No	
Does your product provide special analysis for parallelization libraries?	No	
Which libraries does your product support?	☐ MPI ☐ Intel Threading Building Blocks ☐ OpenCL ☐ CUDA ☐ Other	☐ OpenMP ☐ DirectX Compute ☐ ATI Stream

5.2 Akka, Scalable Solutions AB

Company Information

Vendor	Scalable Solutions
Established in	2003

Headquarters	Backvagen 16B 75652 Uppsala Sweden
Telephone	
Fax	
Website	http://scalablesolutions.se
Summary	Product development, consulting and training in scalable system development.

German Service Partner	No

MWARE multicore software

General Product Information

Product name	Akka
Market entry	2009
Version	0.7
Description	Simpler Scalability, Fault-Tolerance, Concurrency & Remoting through Actors.

Product categories	☐ Profiling	☐ Debugging
	☐ Tuning	☐ CASE / Software Modeling
	☐ Algorithm	☒ Runtime
	☒ Framework	☒ Compiler
	☒ Library	☐ Programming language
	☐ Other	
Product markets	☐ Embedded software	☒ Networking software
	☐ Numerical simulations	☐ Office applications
	☐ Technical applications	☒ High Performance Computing
	☒ Computer Games	☐ Other

Customer references	ORNL, CEA, NERSC, TACC

Software Development

Software development process model	☒ All ☐ V model ☐ Waterfall ☐ Agile: ☐ Other
Recommended stages	Design, Implementation and Deploy
Other related products	
Product combination	

Functionality

Supported installation architectures	☒ x86 ☒ x86_64 ☒ ia64 ☐ ARM ☒ Sparc ☒ PowerPC ☐ Cell Broadband Engine ☐ GPUs: ☐ Other
Supported operating systems	☒ Windows: XP, Vista, 7 ☒ Linux: Debian, Fedora, Mandriva, RedHat, SUSE

	☒ AIX	☒ HP-UX
	☒ BSD: FreeBSD	☒ Solaris
	☒ Mac OS X	☐ Other

Application type	☐ Graphical application	☒ Commandline application
	☐ Plugin: Visual Studio	☒ Library/Framework
	☐ Other	

Trial version	Yes
Time limit	Open Source. Forever.
Limitations	None

Implementation

What programming concepts are fundamental to your product?	☒ Threads (native threads, green threads, n:m threads, etc.)
	☒ Message passing
	☒ Dataflow concepts
	☒ Functional programming concepts
	☐ Asynchronous computation with callbacks
	☐ Lock-free data structures

	☐ Parallel patterns (e.g. Map/Reduce) ☐ Other/Further concepts: ☐ Can't answer
Which memory models form the basis of your product?	☒ Shared memory ☒ Distributed memory ☐ Non-uniform memory access (NUMA)
What special knowledge does a developer/programmer have to have in order to use your product?	☐ Thread programming ☐ Functional programming ☐ Dataflow programming ☐ Parallel patterns ☐ Other
How does a developer model/express parallelism with your product?	☐ Via threads ☐ Via library calls ☐ Via callbacks (filling out a framework) ☒ Via actors (process, agents, etc.) and message passing ☐ Via following a parallel pattern ☐ Other
How can the correctness of the parallel implementation be reviewed?	☒ Debugging using standard tools ☐ Product offers additional, more suitable tools
What other tools do you suggest for reviewing a parallel implementation that uses your product?	

MWARE multicore software

What programming languages does your product support?	☐ All, product is language-independent ☐ Product is a programming language ☐ C/C++ ☒ Java ☐ C# ☐ Fortran ☒ Other: Scala
What features does your product offer for preventing/minimizing common errors in parallel programming?	☐ Deadlock detection ☐ Race condition detection ☒ Transactional memory ☐ Lock-free data structures ☒ Functional semantics ☒ Dataflow semantics ☐ Other
What features of your product make it particularly suitable for the efficient programming of multicore processors?	– Actors (Erlang-style) – STMTransactors (Actors + STM) – Dataflow (Oz-style) – Agents (Clojure-style)
What features help developers get started using your product?	☒ Known/Familiar programming language ☒ Known abstractions ☐ Provision of infrastructure (parallel runtime, parallel memory model, etc.) ☐ Plugins for development environments ☐ Specialized development environments ☐ Other

What products do you recommend for the analysis (profiling, debugging, etc.) of software that was implemented using your product?	Standard Java profilers such as YourKit, JProfiler etc.
What documentation exists for your product?	☒ Meaningful API documentation ☒ Beginner tutorials ☒ Usage examples ☒ Other: Complete reference
What kinds of support do you offer for your product?	☒ On-site support ☒ Phone support ☒ Email support ☒ Conferences ☒ Active community (website, wiki, mailing list, newsgroup, etc.) ☒ Seminars/Workshops ☐ Other

5.3 Allinea DDT, Allinea Software

Company Information

Vendor	Allinea Software Ltd.
Established in	2002

Headquarters	Innovation Centre, Technology Park CV34 6UW, Warwick UK
Telephone	+44 1926 623231
Fax	
Website	www.allinea.com
Summary	Based in Warwick (UK), with subsidiaries in the US and Germany, Allinea Software Ltd. is a leading supplier of tools for parallel programming and high performance computing (HPC). Allinea's products are used by leading commercial and research institutions across the world, and have consistently set the standard for affordability, functionality and ease-of-use – whether applied to applications at modest scale or peta-scale applications on the world's largest supercomputers. With new product features aimed at multi-threaded applications and novel computing architectures, Allinea is now bringing its wealth of experience in parallel tools to the rapidly-expanding arena of multicore processing.

German Service Partner	SMB Kortumstraße 41 448787 Bochum Germany
Telephone	+49 234 5169920
Fax	+49 234 51699229
Website	http://www.developers-source.de

General Product Information

Product name	Allinea DDT - Distributed Debugging Tool
Market entry	2003
Version	2.5
Description	Allinea DDT is a comprehensive graphical debugger for scalar, multi-threaded and large-scale parallel applications that are written in C, C++ and Fortran. The advanced source code browser shows at a glance the state of the processes within a parallel job, and enormously simplifies the task of debugging large numbers of simultaneous processes. DDT has a wealth of features designed to debug effectively - from deadlock and memory leak tools, to data comparison and groupwise process control, and it interoperates with all known MPI implementations and all batch queuing systems. Native CUDA support has just been introduced

Product categories	☒ Profiling	☒ Debugging
	☒ Tuning	☐ CASE / Software Modeling
	☐ Algorithm	☐ Runtime
	☐ Framework	☐ Compiler
	☐ Library	☐ Programming language
	☐ Other	
Product markets	☒ Embedded software	☒ Networking software
	☒ Numerical simulations	☐ Office applications
	☒ Technical applications	☒ High Performance Computing
	☒ Computer Games	☐ Other

Customer references	All parallel application which need to scale up. ORNL, CEA, NERSC and TACC are good customers examples.

Software Development

Software development process model	☐ All	
	☐ V model	☐ Waterfall
	☐ Agile:	☐ Other
Recommended stages	Debugging and optimising codes	

Other related products	Allinea OPT
Related products in combination	Allinea OPT is the awaited companion product to DDT which aims to enable developers and benchmarkers to find bottlenecks in MPI codes and consequently to improve performance. OPT works transparently on local or remote compute resources with a client/server architecture. It is designed to help users improve the performance of parallel codes - for C, C++ and Fortran MPI applications including Massive Parallel Processing (MPP) codes. OPT sets a new standard for the problem of performance analysis and optimization of parallel codes. Being truly GRID capable, OPT allows users to access remote profiling data almost as rapidly as a local server. A top down approach to optimization allows users to concentrate their effort on the parts of their code that really impact overall performance. The novel callgraph capability analyzes codes for low performance and automatically highlights the worst functions. Statistical analysis shows areas of poor communication or computation imbalance across processors. OPT's database of performance sessions allows performance comparison between jobs, between separate applications or different hardware to be made. An impressive set of charts allows scalability to be analyzed effectively.

Functionality

Supported installation architectures	☒ x86	☒ x86_64
	☒ ia64	☐ ARM
	☒ Sparc	☒ PowerPC
	☒ Cell Broadband Engine	☒ GPUs: NVidia
	☐ Other	

Supported operating systems	☒ Windows: XP, Vista, 7	
	☒ Linux: Debian, Fedora, Mandriva, RedHat, SUSE	
	☒ AIX	☐ HP-UX
	☐ BSD: FreeBSD	☒ Solaris
	☐ Mac OS X	☐ Other

Application type	☒ Graphical application	☐ Commandline application
	☐ Plugin: Visual Studio	☐ Library/Framework
	☐ Other	

Trial version	Yes
Time limit	30 days
Limitations	Limited number of MPI processes to debug or to optimise simultanously (16) 1 user

MWARE multicore software

Analysis

Supported analysis architectures	☒ All the architectures the product runs on	
	☐ x86	☐ x86_64
	☐ ARM	☐ Sparc
	☐ PowerPC	☐ Cell Broadband Engine
	☐ GPUs	☐ Other
Supported languages	☒ C	☒ C++
	☒ Fortran	☐ Java
	☐ C#	☐ Python
	☒ .NET	☐ OpenCL
	☐ DirectX Compute	☐ ATI Stream
	☒ Other: CUDA for Allinea DDT	

Analysis level	☐ On the model level (e.g. UML verification, process checking, sequence diagram analysis)
	☒ On the source code level (e.g. static analysis)
	☒ On the communication level (e.g. between processes/threads in MPI)
	☒ On the instruction level (e.g. runtime analysis, profiling)
	☐ Other/Further levels

Analysis preparation	☐ Restart in analysis environment ☒ Compilation with special parameters: Debug parameters for allinea DDT, gdb support (-g), Library Linking for OPT ☒ Manual modifications: ☒ Adding compiler directives ☐ Adding library calls ☐ Other

Analysis results are stored	Yes

Presentation of analysis results	☐ Textual summary ☒ Graphical presentation ☐ Other

Analysis practice	☒ ☐ ☐ ☐ ☐ Very little A lot
Special knowledge required for analysis	No

Analysis results mapping to code	N/A

Automated advice	No
Availability of results	☒ During the analysis ☐ After stopping the analysis ☒ After stopping the analysis and preparing the results
Comparison of analysis runs	Yes
Processing of analysis results is possible with other tools	No

Analysis overhead	2 - 3%
Analysis limit	No
Presentation limit	No

Profiling

Profiling type	☐ System-wide analysis (like DTrace or oprofile) ☒ Software-specific analysis (like gprof) ☒ Library-specific analysis (like TAU for MPI) ☒ MPI ☐ OpenMP ☐ Intel Threading Building Blocks ☐ OpenCL ☐ DirectX Compute ☒ CUDA ☐ ATI Stream ☐ Other ☒ Other: PAPI
Profiling method	☒ Statistical sampling ☒ Instrumentation ☐ Simulation ☒ Event-based ☐ Other
Gathered profiling information	☒ Combined runtimes (flat profile) ☒ Runtimes with callgraph ☒ Runtimes with callgraph and trace

Usage of hardware performance counters	Yes
Multicore analysis	☒ CPU load per thread/process ☐ Memory profiling ☐ I/O profiling ☒ Cache profiling ☐ Other
Analysis of processor idle time	Yes
Measured idle times	☒ Waiting for I/O ☒ Waiting for thread synchronization ☒ Waiting for acquiring locks ☐ Other

Debugging

What kinds of analysis does your product provide?	☒ Static analysis of source code ☐ Runtime analysis ☐ Runtime analysis with special support for certain libraries ☐ Other

How does your product perform runtime analysis?	☐ Instrumentation ☐ Event-based ☐ Simulation ☐ Other
Which libraries does your product support?	☐ MPI ☐ OpenMP ☐ Intel Threading Building Blocks ☐ OpenCL ☐ DirectX Compute ☐ CUDA ☐ ATI Stream ☐ Other
What kinds of parallelization/multi-threading errors does your product look for?	☒ Deadlocks ☒ Livelocks ☐ Race Conditions ☒ Memory leaks ☒ Other: Described in documentation at http://www.allinea.com/DDT.pdf
For what kinds of errors does your product provide a mapping to the causing source code?	☒ For all of the above ☐ Deadlocks ☐ Livelocks ☐ Race Conditions ☐ Memory leaks
Does your product use hardware performance counters during analysis?	Yes
Does your product generate software traces for later analysis?	Yes

In what format are the traces stored?	☒ In a proprietory format	☐ Open Trace Format
	☐ NOPE format	☒ Other
Can your product analyse existing software traces?	No	
What trace formats does your product support?	☐ Open Trace Format	☐ NOPE format
	☐ Other	

Tuning

What kinds of characteristics does your product analyze for performance evaluation?	☒ Processor/Core load	☐ Memory access
	☒ Cache access	☐ Thread communication
	☐ False Sharing	☐ Other
What kinds of techniques does your product use for analysis?	☒ Statistical sampling	☒ Instrumentation
	☐ Simulation	☐ Event-based analysis
	☐ Other	
Does your product use hardware performance counters during analysis?	Yes	
Does your product provide special analysis for parallelization	Yes	

libraries?	
Which libraries does your product support?	☒ MPI ☐ OpenMP ☐ Intel Threading Building Blocks ☐ OpenCL ☐ DirectX Compute ☐ CUDA ☐ ATI Stream ☐ Other

5.4 Pervasive DataRush, Pervasive Software

Company Information

Vendor	Pervasive Software
Established in	1985

Headquarters	12365-B Riata Trace Pkwy 78727 Austin United States
Telephone	512-231-6000
Fax	
Website	www.pervasivedatarush.com
Summary	Pervasive Software (NASDAQ: PVSW) helps companies get the most out of their data investments through embeddable data management, agile data integration software and revolutionary next generation analytics. The embeddable Pervasive PSQL database engine allows organizations to successfully embrace new technologies while maintaining application compatibility and robust database reliability in a near-zero database administration environment. Pervasive's multi-purpose data integration platform accelerates the sharing of information between multiple data stores, applications, and hosted business systems and allows customers to re-use the same software for diverse integration scenarios. Pervasive DataRush is an embeddable high-performance software platform for data intensive processing applications such as claims processing, risk analysis, fraud detection, data mining, predictive analytics, sales optimization and marketing analytics. For more than two decades, Pervasive products have delivered value to tens of

	thousands of customers in more than 150 countries with a compelling combination of performance, flexibility, reliability and low total cost of ownership. Through Pervasive Innovation Labs, the company also invests in exploring and creating cutting edge solutions for the toughest data analysis and data delivery challenges.

German Service Partner	No

General Product Information

Product name	Pervasive DataRush
Market entry	2009
Version	4.2
Description	Pervasive DataRush is a high-performance, embeddable software platform for the development of the next generation of data-intensive processing and analytics. DataRush fully utilizes the power of multicore processors and can process data at unprecedented speeds. Pervasive DataRush hides the complexity of parallel development and gives you the power of a cluster in a single SMP node.

Product categories	☒ Profiling	☒ Debugging
	☐ Tuning	☐ CASE / Software Modeling
	☒ Algorithm	☒ Runtime
	☒ Framework	☐ Compiler

	☒ Library	☒ Programming language
	☐ Other	
Product markets	☐ Embedded software	☒ Networking software
	☒ Numerical simulations	☐ Office applications
	☒ Technical applications	☒ High Performance Computing
	☐ Computer Games	☐ Other

Customer references	

Software Development

Software development process model	☒ All	
	☐ V model	☐ Waterfall
	☐ Agile:	☐ Other
Recommended stages	All stages of software development.	
Other related products	– Pervasive DataMatcher – Pervasive RushRecommender – Pervasive Operator Libraries – Pervasive Data Profiler	
Related products in combination	The above pre-packaged products can be used by the developer in conjunction with Pervasive DataRush for rapid design,	

	development, and testing.

Functionality

Supported installation architectures	☒ x86	☒ x86_64
	☒ ia64	☐ ARM
	☒ Sparc	☒ PowerPC
	☐ Cell Broadband Engine	☐ GPUs:
	☒ Other: Pervasive DataRush runs on any Java 6 JVM.	
Supported operating systems	☒ Windows: XP, Vista, 7	
	☒ Linux: Debian, Fedora, Mandriva, RedHat, SUSE	
	☒ AIX	☒ HP-UX
	☒ BSD: FreeBSD	☒ Solaris
	☒ Mac OS X	☐ Other

Application type	☐ Graphical application	☒ Commandline application
	☐ Plugin: Visual Studio	☒ Library/Framework
	☐ Other	

Trial version	Yes
Time limit	Evaluation period can be negotiated.

Limitations	None

Analysis

Supported analysis architectures	☒ All the architectures the product runs on	
	☐ x86	☐ x86_64
	☐ ARM	☐ Sparc
	☐ PowerPC	☐ Cell Broadband Engine
	☐ GPUs	☐ Other
Supported languages	☐ C	☐ C++
	☐ Fortran	☐ Java
	☐ C#	☐ Python
	☐ .NET	☐ OpenCL
	☐ DirectX Compute	☐ ATI Stream
	☒ Other: Pervasive DataRush Dataflow Language Analysis	

Analysis level	☒ On the model level (e.g. UML verification, process checking, sequence diagram analysis)
	☒ On the source code level (e.g. static analysis)
	☒ On the communication level (e.g. between processes/threads in MPI)
	☐ On the instruction level (e.g. runtime analysis, profiling)

	☐ Other/Further levels

Analysis preparation	☒ Restart in analysis environment ☐ Compilation with special parameters: ☐ Manual modifications: ☐ Adding compiler directives ☐ Adding library calls ☐ Other

Analysis results are stored	No

Analysis results	☐ Textual summary ☒ Graphical presentation ☐ Other

Required analysis practice	☐ ☒ ☐ ☐ ☐ Very little A lot
Special knowledge required for analysis	No

Analysis results mapping to code	Yes
Automated advice	No
Availability of results	☒ During the analysis ☐ After stopping the analysis ☐ After stopping the analysis and preparing the results
Comparison of analysis runs	Yes
Processing of analysis results is possible with other tools	Yes: Any tool that can process Java.

Analysis overhead	Low overhead
Analysis limit	No
Presentation limit	No

MWARE multicore software

Profiling

Profiling type	☐ System-wide analysis (like DTrace or oprofile) ☐ Software-specific analysis (like gprof) ☒ Library-specific analysis (like TAU for MPI) ☐ MPI ☐ OpenMP ☐ Intel Threading Building Blocks ☐ OpenCL ☐ DirectX Compute ☐ CUDA ☐ ATI Stream ☒ Other ☐ Other
Profiling method	☒ Statistical sampling ☐ Instrumentation ☐ Simulation ☐ Event-based ☐ Other
Gathered profiling information	☐ Combined runtimes (flat profile) ☒ Runtimes with callgraph ☐ Runtimes with callgraph and trace

Usage of hardware performance counters	No
Multicore analysis	☒ CPU load per thread/process ☒ Memory profiling ☒ I/O profiling ☐ Cache profiling ☐ Other
Analysis of processor idle time	Yes
Measured times	☒ Waiting for I/O ☒ Waiting for thread synchronization ☐ Waiting for acquiring locks ☐ Other

Debugging

What kinds of analysis does your product provide?	☐ Static analysis of source code ☐ Runtime analysis ☒ Runtime analysis with special support for certain libraries ☐ Other

How does your product perform runtime analysis?	☐ Instrumentation	☐ Event-based
	☐ Simulation	☐ Other
Which libraries does your product support?	☐ MPI	☐ OpenMP
	☐ Intel Threading Building Blocks	
	☐ OpenCL	☐ DirectX Compute
	☐ CUDA	☐ ATI Stream
	☒ Other: Pervasive DataRush and other Java libraries.	
What kinds of parallelization/multi-threading errors does your product look for?	☒ Deadlocks	☒ Livelocks
	☒ Race Conditions	☒ Memory leaks
	☐ Other	
For what kinds of errors does your product provide a mapping to the causing source code?	☐ For all of the above	
	☐ Deadlocks	☐ Livelocks
	☐ Race Conditions	☐ Memory leaks
Does your product use hardware performance counters during analysis?	No	
Does your product generate software traces for later analysis?	Yes	

In what format are the traces stored?	☐ In a proprietory format ☐ Open Trace Format ☐ NOPE format ☒ Other
Can your product analyse existing software traces?	Yes
What trace formats does your product support?	☐ Open Trace Format ☐ NOPE format ☒ Other: Visual VM Application Graph

Implementation

What programming concepts are fundamental to your product?	☐ Threads (native threads, green threads, n:m threads, etc.) ☐ Message passing ☒ Dataflow concepts ☒ Functional programming concepts ☐ Asynchronous computation with callbacks ☒ Lock-free data structures ☒ Parallel patterns (e.g. Map/Reduce) ☐ Other/Further concepts: ☐ Can't answer
Which memory models form the basis of your product?	☒ Shared memory ☒ Distributed memory ☒ Non-uniform memory access (NUMA)

What special knowledge does a developer/programmer have to have in order to use your product?	☐ Thread programming ☐ Functional programming ☒ Dataflow programming ☒ Parallel patterns ☐ Other
How does a developer model/express parallelism with your product?	☐ Via threads ☐ Via library calls ☐ Via callbacks (filling out a framework) ☐ Via actors (process, agents, etc.) and message passing ☒ Via following a parallel pattern ☐ Other
How can the correctness of the parallel implementation be reviewed?	☒ Debugging using standard tools ☒ Product offers additional, more suitable tools
What other tools do you suggest for reviewing a parallel implementation that uses your product?	VisualVM plug-in
What programming languages does your product support?	☐ All, product is language-independent ☐ Product is a programming language ☐ C/C++ ☒ Java ☐ C# ☐ Fortran ☐ Other

What features does your product offer for preventing/minimizing common errors in parallel programming?	☒ Deadlock detection ☐ Transactional memory ☐ Functional semantics ☐ Other ☒ Race condition detection ☒ Lock-free data structures ☒ Dataflow semantics
What features of your product make it particularly suitable for the efficient programming of multicore processors?	Pervasive DataRush is a 100% Java framework that allows developers to quickly build highly parallel, data-intensive applications that take full advantage of multicore, SMP platforms. No specialized knowledge is required in threading, concurrent memory access, deadlock detection, data workload partitioning/buffering or any other complex aspect of parallel thread execution. In fact, now developers can quickly build highly-parallel data processing applications for today\'s multicore hardware all without the need to deal with threading libraries, deadlock detection algorithms, or concurrent process design issues.Pervasive DataRush comes with a rich library of out-of-the-box Java components that can be assembled into a series of dataflow operations. Where custom components need to be added or extended, developers simply use the Pervasive DataRush SDK to quickly build and extend their Pervasive DataRush application.
What features help developers get started using your product?	☒ Known/Familiar programming language ☒ Known abstractions ☐ Provision of infrastructure (parallel runtime, parallel memory model, etc.) ☒ Plugins for development environments ☐ Specialized development environments ☐ Other
What products do you recommend for the analysis (profiling, debugging, etc.) of software that was	Any Java 6 IDE

implemented using your product?	
What documentation exists for your product?	☒ Meaningful API documentation ☒ Beginner tutorials ☒ Usage examples ☐ Other
What kinds of support do you offer for your product?	☒ On-site support ☒ Phone support ☒ Email support ☒ Conferences ☒ Active community (website, wiki, mailing list, newsgroup, etc.) ☒ Seminars/Workshops ☐ Other

5.5 Enterprise Architect, SparxSystems Central Europe

Company Information

Vendor	SparxSystems Central Europe
Established in	1998

Headquarters	Handelskay 340 1020 Wien Austria
Telephone	+43 (0)662 90 600 2041
Fax	+43 (0)662 90 333 3041
Website	www.sparxsystems.de
Summary	SparxSystems is an Australian Company providing a conceptual modeling tool, especially for UML. SparxSystems CE is the European sister company, responsible for licensing, training, and consulting.

German Service Partner	N/A

MWARE multicore software

General Product Information

Product name	Enterprise Architect
Market entry	1998
Version	8.0
Description	EA is a conceptual modeling tool which provides tools for the whole team. It supports requirement engineering, analysis, design, and implementation models, as well as round trip engineering.

Product categories	☐ Profiling	☐ Debugging
	☐ Tuning	☒ CASE / Software Modeling
	☐ Algorithm	☐ Runtime
	☐ Framework	☐ Compiler
	☐ Library	☐ Programming language
	☐ Other	
Product markets	☒ Embedded software	☐ Networking software
	☐ Numerical simulations	☒ Office applications
	☒ Technical applications	☐ High Performance Computing
	☐ Computer Games	☒ Other: EA can also be used for Business Modelling. It supports all common modeling languages like: SysML, SoaML, BPMN, etc.

Software Development

Software development process model	☒ All --- ☐ V model ☐ Waterfall ☐ Agile: ☐ Other
Recommended stages	Analyse, Design, Implementation
Other related products	

Functionality

Supported installation architectures	☒ x86 ☐ x86_64 ☐ ia64 ☐ ARM ☐ Sparc ☐ PowerPC ☐ Cell Broadband Engine ☐ GPUs: ☐ Other
Supported operating systems	☒ Windows: XP, Vista, 7 ☐ Linux: Debian, Fedora, Mandriva, RedHat, SUSE ☐ AIX ☐ HP-UX ☐ BSD: FreeBSD ☐ Solaris ☐ Mac OS X ☒ Other: Mac OS and Linux are supported via Crossover

Application type	☒ Graphical application	☐ Commandline application
	☐ Plugin: Visual Studio	☐ Library/Framework
	☒ Other: EA can be accessed via API	

Trial version	N/A

Implementation

What programming concepts are fundamental to your product?	☐ Threads (native threads, green threads, n:m threads, etc.) ☐ Message passing ☐ Dataflow concepts ☐ Functional programming concepts ☐ Asynchronous computation with callbacks ☐ Lock-free data structures ☐ Parallel patterns (e.g. Map/Reduce) ☐ Other/Further concepts: ☐ Can't answer
Which memory models form the basis of your product?	☐ Shared memory ☐ Distributed memory ☐ Non-uniform memory access (NUMA)

What special knowledge does a developer/programmer have to have in order to use your product?	☐ Thread programming ☐ Functional programming ☐ Dataflow programming ☐ Parallel patterns ☐ Other
How does a developer model/express parallelism with your product?	☐ Via threads ☐ Via library calls ☐ Via callbacks (filling out a framework) ☐ Via actors (process, agents, etc.) and message passing ☐ Via following a parallel pattern ☐ Other
How can the correctness of the parallel implementation be reviewed?	☐ Debugging using standard tools ☐ Product offers additional, more suitable tools
What other tools do you suggest for reviewing a parallel implementation that uses your product?	
What programming languages does your product support?	☐ All, product is language-independent ☐ Product is a programming language ☐ C/C++ ☐ Java ☐ C# ☐ Fortran ☐ Other

What features does your product offer for preventing/minimizing common errors in parallel programming?	☐ Deadlock detection ☐ Race condition detection ☐ Transactional memory ☐ Lock-free data structures ☐ Functional semantics ☐ Dataflow semantics ☐ Other
What features of your product make it particularly suitable for the efficient programming of multicore processors?	
What features help developers get started using your product?	☐ Known/Familiar programming language ☐ Known abstractions ☐ Provision of infrastructure (parallel runtime, parallel memory model, etc.) ☐ Plugins for development environments ☐ Specialized development environments ☐ Other
What products do you recommend for the analysis (profiling, debugging, etc.) of software that was implemented using your product?	

What documentation exists for your product?	☐ Meaningful API documentation ☐ Beginner tutorials ☐ Usage examples ☐ Other
What kinds of support do you offer for your product?	☐ On-site support ☐ Phone support ☐ Email support ☐ Conferences ☐ Active community (website, wiki, mailing list, newsgroup, etc.) ☐ Seminars/Workshops ☐ Other

5.6 GPI, Fraunhofer ITWM

Company Information

Vendor	Fraunhofer ITWM
Established in	1996

Headquarters	Fraunhofer Platz 1 67663 Kaiserslautern Germany
Telephone	
Fax	
Website	itwm.fraunhofer.de
Summary	Fraunhofer ITWM beschäftigit sich mit numerischen Simulationen und Optimierungen in den Bereichen Fahrzeugtechnik, personalisierte Medizin, robuste und sichere Finanzmärkte, erneuerbare Energien, Stochastik und Homogenität in Prozessen und Materialien.

German Service Partner	N/A

MWARE multicore software

General Product Information

Product name	GPI -- Global address space programming interface
Market entry	2010
Version	2
Description	The GPI is an programming API for the PGAS (Partitioned global address space) programming model. It is designed to give the hardware capabilities directly to the application. It is easy to use, robust and allows for scalable application on todays and tomorrow cluster infrastructure.

Product categories	☐ Profiling	☐ Debugging
	☐ Tuning	☐ CASE / Software Modeling
	☐ Algorithm	☒ Runtime
	☐ Framework	☐ Compiler
	☒ Library	☐ Programming language
	☒ Other: Programming API	
Product markets	☐ Embedded software	☐ Networking software
	☒ Numerical simulations	☐ Office applications
	☒ Technical applications	☒ High Performance Computing
	☐ Computer Games	☐ Other

Customer references	Seismic: GRT Visualization: PsPRO

Software Development

Software development process model	☒ All - ☐ V model ☐ Waterfall ☐ Agile: ☐ Other
Recommended stages	It is a low level tool, best used by high performance experts operating on that level.
Other related products	MCTP -- The multicore thread pool package
Related products in combination	The MCTP is a replacement for Posix threads and overcomes some of their restrictions. It is the only full hardware aware thread package available and beats other packages in benchmarks by at least one order of magnitude in critical operations like synchronization of several threads.Using the MCTP on the node level, you can directly scale your application on the cluster level by using GPI. Both together are the tools for scalable HPC applications.

Functionality

Supported installation architectures	☒ x86	☒ x86_64
	☐ ia64	☐ ARM
	☐ Sparc	☐ PowerPC
	☒ Cell Broadband Engine	☒ GPUs: Any, that allow to read or write to GPU memory via RDMA.
	☐ Other	
Supported operating systems	☐ Windows: XP, Vista, 7	
	☒ Linux: Debian, Fedora, Mandriva, RedHat, SUSE	
	☒ AIX	☒ HP-UX
	☒ BSD: FreeBSD	☒ Solaris
	☒ Mac OS X	☐ Other

Application type	☐ Graphical application	☒ Commandline application
	☐ Plugin: Visual Studio	☒ Library/Framework
	☐ Other	

Trial version	Yes
Time limit	Depends on the customer. Typically 3 months without any limitation
Limitations	None

Implementation

What programming concepts are fundamental to your product?	☒ Threads (native threads, green threads, n:m threads, etc.) ☐ Message passing ☐ Dataflow concepts ☐ Functional programming concepts ☐ Asynchronous computation with callbacks ☒ Lock-free data structures ☒ Parallel patterns (e.g. Map/Reduce) ☒ Other/Further concepts: one sided communication, asynchronous communication ☐ Can't answer
Which memory models form the basis of your product?	☒ Shared memory ☒ Distributed memory ☒ Non-uniform memory access (NUMA)
What special knowledge does a developer/programmer have to have in order to use your product?	☒ Thread programming ☐ Functional programming ☐ Dataflow programming ☒ Parallel patterns ☒ Other: understanding of how to achieve maximal locality
How does a developer model/express parallelism with your product?	☒ Via threads ☐ Via library calls ☐ Via callbacks (filling out a framework) ☐ Via actors (process, agents, etc.) and message passing ☐ Via following a parallel pattern

	☒ Other: via different communication queues (channels), via several communications that happen at the same time, via overlap of computation and communication
How can the correctness of the parallel implementation be reviewed?	☒ Debugging using standard tools ☐ Product offers additional, more suitable tools
What other tools do you suggest for reviewing a parallel implementation that uses your product?	
What programming languages does your product support?	☐ All, product is language-independent ☐ Product is a programming language ☒ C/C++ ☐ Java ☐ C# ☒ Fortran ☐ Other
What features does your product offer for preventing/minimizing common errors in parallel programming?	☐ Deadlock detection ☐ Race condition detection ☐ Transactional memory ☐ Lock-free data structures ☐ Functional semantics ☐ Dataflow semantics ☐ Other
What features of your product make it particularly suitable for the efficient programming of multicore processors?	It is designed for speed with manycore architectures in mind.

What features help developers get started using your product?	☒ Known/Familiar programming language ☒ Known abstractions ☒ Provision of infrastructure (parallel runtime, parallel memory model, etc.) ☐ Plugins for development environments ☐ Specialized development environments ☒ Other: Documents such as 'Getting started', tutorials, extensive interface documentation
What products do you recommend for the analysis (profiling, debugging, etc.) of software that was implemented using your product?	Standard tools: – Gdb – Vampir – DDT – Valgrind
What documentation exists for your product?	☒ Meaningful API documentation ☒ Beginner tutorials ☒ Usage examples ☐ Other
What kinds of support do you offer for your product?	☐ On-site support ☐ Phone support ☒ Email support ☐ Conferences ☐ Active community (website, wiki, mailing list, newsgroup, etc.) ☐ Seminars/Workshops ☐ Other

5.7 IESE Simulink partitioner, Fraunhofer IESE

Company Information

Vendor	Fraunhofer IESE
Established in	

Headquarters	Fraunhofer-Platz 1 67663 Kaiserslautern Germany
Telephone	
Fax	
Website	www.iese.fraunhofer.de
Summary	

German Service Partner	N/A

MWARE multicore software

General Product Information

Product name	IESE Simulink partitioner
Market entry	2010
Version	1.0
Description	The product partitions and optimizes Simulink models for efficient execution on multicore platforms

Product categories	☐ Profiling	☐ Debugging
	☒ Tuning	☒ CASE / Software Modeling
	☐ Algorithm	☒ Runtime
	☐ Framework	☐ Compiler
	☐ Library	☐ Programming language
	☐ Other	
Product markets	☒ Embedded software	☐ Networking software
	☐ Numerical simulations	☐ Office applications
	☐ Technical applications	☐ High Performance Computing
	☐ Computer Games	☐ Other

Customer references	

Software Development

Software development process model	☒ All --- ☐ V model ☐ Waterfall ☐ Agile: ☐ Other
Recommended stages	During Implementation/Deployment activities
Other related products	– Technology consulting – Scoping – Optimization and parallelization of native code – Architecture analysis
Related products in combination	Products/services cover all stages of most process models

Functionality

Supported installation architectures	☒ x86 ☒ x86_64 ☐ ia64 ☒ ARM ☐ Sparc ☐ PowerPC ☐ Cell Broadband Engine ☐ GPUs: ☒ Other: FPGA based processors (Microblaze)

MWARE multicore software

Supported operating systems	☐ Windows: XP, Vista, 7	
	☐ Linux: Debian, Fedora, Mandriva, RedHat, SUSE	
	☐ AIX	☐ HP-UX
	☐ BSD: FreeBSD	☐ Solaris
	☐ Mac OS X	☒ Other: Runs on-top of embedded or customer specific operating systems, e.g. FreeRTOS, VxWorks

Application type	☐ Graphical application	☐ Commandline application
	☐ Plugin: Visual Studio	☐ Library/Framework
	☒ Other: Part of consulting service	

Trial version	N/A

Analysis

Supported analysis architectures	☐ All the architectures the product runs on	
	☒ x86	☒ x86_64
	☐ ARM	☐ Sparc
	☐ PowerPC	☐ Cell Broadband Engine
	☐ GPUs	☐ Other

Supported languages	☐ C ☐ C++ ☐ Fortran ☐ Java ☐ C# ☐ Python ☐ .NET ☐ OpenCL ☐ DirectX Compute ☐ ATI Stream ☒ Other: Simulink / generic data-flow based languages

Analysis level	☒ On the model level (e.g. UML verification, process checking, sequence diagram analysis) ☒ On the source code level (e.g. static analysis) ☐ On the communication level (e.g. between processes/threads in MPI) ☐ On the instruction level (e.g. runtime analysis, profiling) ☐ Other/Further levels

Analysis preparation	☐ Restart in analysis environment ☐ Compilation with special parameters: ☐ Manual modifications: ☐ Adding compiler directives ☐ Adding library calls ☐ Other

Analysis results are stored	No

Presentation of analysis results	☒ Textual summary ☐ Graphical presentation ☐ Other

Required analysis practice	☐ ☐ ☒ ☐ ☐ Very little A lot
Special knowledge required for analysis	No

Analysis results mapping to code	No
Automated advice	No
Availability of results	☐ During the analysis ☐ After stopping the analysis ☒ After stopping the analysis and preparing the results
Comparison of analysis runs	No
Processing of analysis results is possible with other tools	No

Analysis overhead	Analysis runs at compile time
Analysis limit	No
Presentation limit	No

Tuning

What kinds of characteristics does your product analyze for performance evaluation?	☒ Processor/Core load	☒ Memory access
	☒ Cache access	☒ Thread communication
	☐ False Sharing	☐ Other
What kinds of techniques does your product use for analysis?	☐ Statistical sampling	☐ Instrumentation
	☐ Simulation	☐ Event-based analysis
	☒ Other: Model checking, semantic analysis, flow analysis	
Does your product use hardware performance counters during analysis?	No	
Does your product provide special analysis for parallelization libraries?	N/A	

Which libraries does your product support?	☐ MPI	☐ OpenMP
	☐ Intel Threading Building Blocks	
	☐ OpenCL	☐ DirectX Compute
	☐ CUDA	☐ ATI Stream
	☐ Other	

Implementation

What programming concepts are fundamental to your product?	☐ Threads (native threads, green threads, n:m threads, etc.)	
	☒ Message passing	
	☒ Dataflow concepts	
	☒ Functional programming concepts	
	☒ Asynchronous computation with callbacks	
	☒ Lock-free data structures	
	☒ Parallel patterns (e.g. Map/Reduce)	
	☐ Other/Further concepts:	
	☐ Can't answer	
Which memory models form the basis of your product?	☒ Shared memory	
	☒ Distributed memory	
	☒ Non-uniform memory access (NUMA)	
What special knowledge does a developer/programmer have to have in order	☐ Thread programming	☐ Functional programming
	☒ Dataflow programming	☐ Parallel patterns

to use your product?	☒ Other: Platform knowledge
How does a developer model/express parallelism with your product?	☐ Via threads ☐ Via library calls ☐ Via callbacks (filling out a framework) ☐ Via actors (process, agents, etc.) and message passing ☐ Via following a parallel pattern ☒ Other: Parallelism is derived from data-flow specification
How can the correctness of the parallel implementation be reviewed?	☒ Debugging using standard tools ☐ Product offers additional, more suitable tools
What other tools do you suggest for reviewing a parallel implementation that uses your product?	
What programming languages does your product support?	☐ All, product is language-independent ☐ Product is a programming language ☐ C/C++ ☐ Java ☐ C# ☐ Fortran ☒ Other: Simulink, ASCET
What features does your product offer for preventing/minimizing common errors in parallel programming?	☐ Deadlock detection ☒ Race condition detection ☐ Transactional memory ☒ Lock-free data structures ☐ Functional semantics ☒ Dataflow semantics

	☐ Other
What features of your product make it particularly suitable for the efficient programming of multi-core processors?	– Developers do not need to know about parallelization – Existing algorithms may be re-used – Additional front-ends may be created for integrating additional languages – Users of the product need to provide model of platform and network on chip via tailored DSL – Platform experts compose tailored runtime environment out of provided components
What features help developers get started using your product?	☐ Known/Familiar programming language ☐ Known abstractions ☐ Provision of infrastructure (parallel runtime, parallel memory model, etc.) ☐ Plugins for development environments ☒ Specialized development environments ☐ Other
What products do you recommend for the analysis (profiling, debugging, etc.) of software that was implemented using your product?	Simulink Real-Time Workshop
What documentation exists for your product?	☐ Meaningful API documentation ☐ Beginner tutorials ☐ Usage examples ☒ Other: Internal reference documentation and reports

What kinds of support do you offer for your product?	☐ On-site support	☐ Phone support
	☐ Email support	☐ Conferences
	☐ Active community (website, wiki, mailing list, newsgroup, etc.)	
	☐ Seminars/Workshops	☒ Other: Product is intended for in-house use as part of consulting services for now

5.8 Intel® Ct Technology, Intel Corporation

Company Information

Vendor	Intel Corporation
Established in	1968

Headquarters	2200 Mission College Boulevard Santa Clara, CA 95054 United States of America
Telephone	
Fax	
Website	www.intel.com
Summary	Intel Corporation is a semiconductor chip maker, developing advanced integrated digital technology products, primarily integrated circuits, for industries, such as computing and communications. The Company designs and manufactures computing and communications components, such as microprocessors, chipsets, motherboards, and wireless and wired connectivity products, as well as platforms that incorporate these components. It operates in nine operating segments: PC Client Group, Data Center Group, Embedded and Communications Group, Digital Home Group, Ultra-Mobility Group, NAND Solutions Group, Wind River Software Group, Software and Services Group and Digital Health Group.

German Service Partner	No

General Product Information

Product name	Intel(R) Ct Technology
Market entry	Private Beta 1: Q4 2009
Version	Private Beta 2
Description	Intel's Ct technology provides a generalized data parallel programming solution that frees application developers from dependencies on particular low-level parallelism mechanisms or hardware architectures. It produces scalable, portable, and deterministic parallel implementations from a single high-level, maintainable, and application-oriented specification of the desired computation. It is ideal for applications that require data-intensive mathematical computations such as those found in medical imaging, digital content creation, financial analytics, energy, data mining, science and engineering. Ct technology integrates with and complements other Intel developer and parallel programming tools. Applications written today with Intel's Ct technology will automatically scale to support tomorrow's multicore and manycore processors, thereby protecting your development investment.

Product categories	☐ Profiling	☐ Debugging
	☐ Tuning	☐ CASE / Software Modeling
	☐ Algorithm	☒ Runtime

	☒ Framework ☒ Compiler ☒ Library ☒ Programming language ☒ Other: Intel (R) Ct Technology is both a library/framework and a language to specify an arbitrary parallel computation.
Product markets	☐ Embedded software ☐ Networking software ☒ Numerical simulations ☐ Office applications ☒ Technical applications ☒ High Performance Computing ☒ Computer Games ☒ Other: Bioinformatics, engineering design, financial analytics, oil and gas, medical imaging, visual computing, machine learning and AI, climate and weather simulation, planetary exploration and astrophysics, signal and image processing, enterprise applications, database search

Customer references	Customer evaluations for Ct in progress.

Software Development

Software development process model	☒ All --- ☐ V model ☐ Waterfall ☐ Agile: ☐ Other

Recommended stages	Prototype and delivery code development
Other related products	– Intel(R) Threading Building Blocks – Intel (R) Math Kernel Library – Intel (R) Integrated Performance Primitives – Intel (R) MPI – Intel (R) Compiler Professional Edition and Suite Edition – Intel(R) VTune – Intel (R) Parallel Studio – Intel (R) Thread Profiler – Intel (R) Thread Checker – Intel (R) Cluster Tools – Intel (R) Trace Analyzer and Collector.
Related products in combination	Ct Integrates with existing IDEs, tools, and compilers In addition to working seamlessly with other Intel parallel programming tools (e.g. Intel® Math Kernel Library (Intel® MKL))

Functionality

Supported installation architectures	☒ x86	☒ x86_64
	☒ ia64	☐ ARM
	☐ Sparc	☐ PowerPC
	☐ Cell Broadband Engine	☐ GPUs:
	☒ Other: Support for other processors will probably be added in the future.	
Supported operating systems	☒ Windows: XP, Vista, 7	
	☒ Linux: Debian, Fedora, Mandriva, RedHat, SUSE	
	☐ AIX	☐ HP-UX

	☐ BSD: FreeBSD	☐ Solaris
	☐ Mac OS X	☒ Other: Support for other operating systems may be added in the future.
Application type	☐ Graphical application	☐ Commandline application
	☐ Plugin: Visual Studio	☒ Library/Framework
	☒ Other: Available as both standalone library and bundled with various software tool products.	

Trial version	Yes
Time limit	Currently available in private beta release. After product shipment, 30 day evaluation will be available.
Limitations	None

Implementation

What programming concepts are fundamental to your product?	☐ Threads (native threads, green threads, n:m threads, etc.)
	☐ Message passing
	☐ Dataflow concepts
	☒ Functional programming concepts
	☐ Asynchronous computation with callbacks
	☐ Lock-free data structures

	☒ Parallel patterns (e.g. Map/Reduce) ☒ Other/Further concepts: Metaprogramming concepts ☐ Can't answer
Which memory models form the basis of your product?	☒ Shared memory ☒ Distributed memory ☒ Non-uniform memory access (NUMA)
What special knowledge does a developer/programmer have to have in order to use your product?	☐ Thread programming ☐ Functional programming ☐ Dataflow programming ☒ Parallel patterns ☐ Other
How does a developer model/express parallelism with your product?	☐ Via threads ☐ Via library calls ☐ Via callbacks (filling out a framework) ☐ Via actors (process, agents, etc.) and message passing ☐ Via following a parallel pattern ☒ Other: We add new types and operations to the existing C++ standard. They then use those types and operations to create a kernel of computation that we then optimize at runtime to run on multiple cores through both vectorization and threading. These kernels can be used inside multiple parallel patterns.
How can the correctness of the parallel implementation be reviewed?	☒ Debugging using standard tools ☒ Product offers additional, more suitable tools
What other tools do you suggest for reviewing a parallel	Data visualization tools are also provided.

implementation that uses your product?	
What programming languages does your product support?	☐ All, product is language-independent ☒ Product is a programming language ☒ C/C++ ☐ Java ☐ C# ☐ Fortran ☒ Other: The Intel (R) Ct Technology has the ability to support any language through a Virtual Machine API. Note: The C++ interface of Ct can be considered a programming language embedded inside C++.
What features does your product offer for preventing/minimizing common errors in parallel programming?	☐ Deadlock detection ☐ Race condition detection ☐ Transactional memory ☐ Lock-free data structures ☒ Functional semantics ☐ Dataflow semantics ☒ Other: Since race conditions cannot be written, they do not have to be debugged. We guarantee determininistic execution equivalent-to-sequential semantics.
What features of your product make it particularly suitable for the efficient programming of multicore processors?	Ct Integrates with existing IDEs, tools, and compilers In addition to working seamlessly with other Intel parallel programming tools (e.g. Intel® Math Kernel Library (Intel® MKL))
What features help developers get started using your product?	☒ Known/Familiar programming language ☒ Known abstractions ☒ Provision of infrastructure (parallel runtime, parallel memory model, etc.)

	☒ Plugins for development environments ☐ Specialized development environments ☐ Other
What products do you recommend for the analysis (profiling, debugging, etc.) of software that was implemented using your product?	– Intel(R) Thread Profiler – Intel(R) VTune(TM) Performance Analyzer
What documentation exists for your product?	☒ Meaningful API documentation ☒ Beginner tutorials ☒ Usage examples ☐ Other
What kinds of support do you offer for your product?	☒ On-site support ☒ Phone support ☒ Email support ☒ Conferences ☒ Active community (website, wiki, mailing list, newsgroup, etc.) ☒ Seminars/Workshops ☐ Other

5.9 JProfiler, ej-technologies GmbH

Company Information

Vendor	ej-technologies
Established in	2001

Headquarters	Claude-Lorrain-Str. 7 81543 München Germany
Telephone	+49 89 65309-151
Fax	+49 89 65309-224
Website	http://www.ej-technologies.com
Summary	ej-technologies develops sophisticated Java tools designed to help programmers make the most of their own applications. With its focused expertise in the areas of performance and deployment and a strong support for open source initiatives, ej-technologies is developing the next generation of enterprise application development tools.

German Service Partner	N/A

MWARE multicore software

General Product Information

Product name	JProfiler
Market entry	2002
Version	6.0
Description	JProfiler is an award-winning all-in-one Java profiler. JProfiler's intuitive GUI helps you find performance bottlenecks, pin down memory leaks and resolve threading issues.

Product categories	☒ Profiling	☒ Debugging
	☐ Tuning	☐ CASE / Software Modeling
	☐ Algorithm	☐ Runtime
	☐ Framework	☐ Compiler
	☐ Library	☐ Programming language
	☐ Other	
Product markets	☐ Embedded software	☒ Networking software
	☒ Numerical simulations	☒ Office applications
	☒ Technical applications	☒ High Performance Computing
	☒ Computer Games	☐ Other

Customer references	http://www.ej-technologies.com/products/jprofiler/testimonials.html

Software Development

Software development process model	☒ All --- ☐ V model ☐ Waterfall ☐ Agile: ☐ Other
Recommended stages	Development, QA, Problems in production
Other related products	No

Functionality

Supported installation architectures	☒ x86 ☒ x86_64 ☒ ia64 ☐ ARM ☒ Sparc ☒ PowerPC ☐ Cell Broadband Engine ☐ GPUs: ☐ Other
Supported operating systems	☒ Windows: XP, Vista, 7 ☒ Linux: Debian, Fedora, Mandriva, RedHat, SUSE ☒ AIX ☒ HP-UX ☐ BSD: FreeBSD ☒ Solaris ☒ Mac OS X ☐ Other

Application type	☒ Graphical application	☒ Commandline application
	☒ Plugin: Visual Studio	☒ Library/Framework
	☐ Other	

Trial version	Yes
Time limit	10 days
Limitations	None, only an evaluation warning is displayed, each time when a profiling session is started.

Analysis

Supported analysis architectures	☒ All the architectures the product runs on	
	☐ x86	☐ x86_64
	☐ ARM	☐ Sparc
	☐ PowerPC	☐ Cell Broadband Engine
	☐ GPUs	☐ Other
Supported languages	☐ C	☐ C++
	☐ Fortran	☒ Java
	☐ C#	☐ Python
	☐ .NET	☐ OpenCL

	☐ DirectX Compute ☐ ATI Stream ☐ Other

Analysis level	☐ On the model level (e.g. UML verification, process checking, sequence diagram analysis) ☐ On the source code level (e.g. static analysis) ☐ On the communication level (e.g. between processes/threads in MPI) ☒ On the instruction level (e.g. runtime analysis, profiling) ☐ Other/Further levels

Analysis preparation	☒ Restart in analysis environment ☐ Compilation with special parameters: ☐ Manual modifications: ☐ Adding compiler directives ☐ Adding library calls ☐ Other

Analysis results are stored	Yes

Presentation of analysis results	☐ Textual summary ☒ Graphical presentation

	☐ Other

Required analysis practice	☐ ☒ ☐ ☐ ☐ Very little A lot
Special knowledge required for analysis	No

Analysis results mapping to code	Yes
Automated advice	No
Availability of results	☒ During the analysis ☒ After stopping the analysis ☐ After stopping the analysis and preparing the results
Comparison of analysis runs	Yes
Processing of analysis results is possible with other tools	No

Analysis overhead	impossible to say in general, depends on many factors, but can be as low as a few percent for certain analyses

Analysis limit	No
Presentation limit	No

Profiling

Profiling type	☐ System-wide analysis (like DTrace or oprofile) ☒ Software-specific analysis (like gprof) ☐ Library-specific analysis (like TAU for MPI) ☐ MPI ☐ OpenMP ☐ Intel Threading Building Blocks ☐ OpenCL ☐ DirectX Compute ☐ CUDA ☐ ATI Stream ☐ Other ☐ Other
Profiling method	☒ Statistical sampling ☒ Instrumentation ☐ Simulation ☒ Event-based ☐ Other

Gathered profiling information	☐ Combined runtimes (flat profile) ☒ Runtimes with callgraph ☒ Runtimes with callgraph and trace
Usage of hardware performance counters	No
Multicore analysis	☒ CPU load per thread/process ☒ Memory profiling ☒ I/O profiling ☐ Cache profiling ☒ Other
Analysis of processor idle time	Yes
Measured times	☒ Waiting for I/O ☒ Waiting for thread synchronization ☒ Waiting for acquiring locks ☐ Other

Debugging

What kinds of analysis does your product provide?	☐ Static analysis of source code	
	☒ Runtime analysis	
	☐ Runtime analysis with special support for certain libraries	
	☐ Other	
How does your product perform runtime analysis?	☒ Instrumentation	☒ Event-based
	☐ Simulation	☐ Other
Which libraries does your product support?	☐ MPI	☐ OpenMP
	☐ Intel Threading Building Blocks	
	☐ OpenCL	☐ DirectX Compute
	☐ CUDA	☐ ATI Stream
	☐ Other	
What kinds of parallelization/multi-threading errors does your product look for?	☒ Deadlocks	☒ Livelocks
	☐ Race Conditions	☒ Memory leaks
	☐ Other	
For what kinds of errors does your product provide a mapping to the causing source code?	☒ For all of the above	
	☐ Deadlocks	☐ Livelocks
	☐ Race Conditions	☐ Memory leaks

Does your product use hardware performance counters during analysis?	No	
Does your product generate software traces for later analysis?	Yes	
In what format are the traces stored?	☒ In a proprietory format	☐ Open Trace Format
	☐ NOPE format	☐ Other
Can your product analyse existing software traces?	No	
What trace formats does your product support?	☐ Open Trace Format	☐ NOPE format
	☐ Other	

5.10 Offload: Community Edition, Codeplay Software Ltd

Company Information

Vendor	Codeplay Software Ltd
Established in	2002

Headquarters	York Place Edinburgh UK
Telephone	44 131 466 0503
Fax	
Website	www.codeplay.com
Summary	Codeplay research and develop powerful compiler technology and software development tools to use with advanced manycore processor architectures.

German Service Partner	No

MWARE multicore software

General Product Information

Product name	Offload: Community Edition
Market entry	2009
Version	1.0.3 beta for Playstation3 and 1.0.1 beta for Cell
Description	Offload is a tool suite and programming model for the easy and efficient Offloading code to SPU\'s on the Cell Processor on Playstation3 and Linux powered Cell devices. The Offload programming model is suitable for and can be ported to all heterogenous multicore processors.

Product categories	☐ Profiling	☒ Debugging
	☐ Tuning	☐ CASE / Software Modeling
	☐ Algorithm	☒ Runtime
	☐ Framework	☒ Compiler
	☒ Library	☒ Programming language
	☐ Other	
Product markets	☐ Embedded software	☐ Networking software
	☐ Numerical simulations	☐ Office applications
	☐ Technical applications	☒ High Performance Computing
	☒ Computer Games	☐ Other

Customer references	

Software Development

Software development process model	☒ All	
	☐ V model	☐ Waterfall
	☐ Agile:	☐ Other
Recommended stages	At beginning or mid stage of a programming project.	
Other related products	No	

Functionality

Supported installation architectures	☒ x86	☐ x86_64
	☐ ia64	☐ ARM
	☐ Sparc	☐ PowerPC
	☐ Cell Broadband Engine	☐ GPUs:
	☐ Other	
Supported operating systems	☒ Windows: XP, Vista, 7	
	☐ Linux: Debian, Fedora, Mandriva, RedHat, SUSE	

	☐ AIX	☐ HP-UX
	☐ BSD: FreeBSD	☐ Solaris
	☐ Mac OS X	☐ Other

Application type	☒ Graphical application	☒ Commandline application
	☐ Plugin: Visual Studio	☒ Library/Framework
	☐ Other	

Trial version	Yes
Time limit	Indefinetely - Offload is free to download and use in commercial projects.
Limitations	Subject to licencing conditions. All commercial projects for which Offload has been used must display credits through a logo splash screen.

Analysis

Supported analysis architectures	☐ All the architectures the product runs on	
	☐ x86	☐ x86_64
	☐ ARM	☐ Sparc
	☐ PowerPC	☒ Cell Broadband Engine

	☐ GPUs	☐ Other
Supported languages	☐ C	☒ C++
	☐ Fortran	☐ Java
	☐ C#	☐ Python
	☐ .NET	☐ OpenCL
	☐ DirectX Compute	☐ ATI Stream
	☐ Other	

Analysis level	☐ On the model level (e.g. UML verification, process checking, sequence diagram analysis)
	☒ On the source code level (e.g. static analysis)
	☐ On the communication level (e.g. between processes/threads in MPI)
	☒ On the instruction level (e.g. runtime analysis, profiling)
	☐ Other/Further levels

Analysis preparation	☒ Restart in analysis environment
	☐ Compilation with special parameters:
	☐ Manual modifications:
	☐ Adding compiler directives ☐ Adding library calls ☐ Other

Analysis results are stored	No

Presentation of analysis results	☒ Textual summary ☐ Graphical presentation ☐ Other

Required analysis practice	☐ ☒ ☐ ☐ ☐ Very little A lot
Special knowledge required for analysis	No

Analysis results mapping to code	No
Automated advice	Yes
Availability of results	☐ During the analysis ☒ After stopping the analysis ☐ After stopping the analysis and preparing the results
Comparison of analysis runs	No

Processing of analysis results is possible with other tools	No

Analysis overhead	N/A
Analysis limit	No
Presentation limit	No

Debugging

What kinds of analysis does your product provide?	☐ Static analysis of source code ☐ Runtime analysis ☐ Runtime analysis with special support for certain libraries ☐ Other	
How does your product perform runtime analysis?	☐ Instrumentation ☐ Simulation	☐ Event-based ☐ Other
Which libraries does your product support?	☐ MPI ☐ Intel Threading Building Blocks ☐ OpenCL ☐ CUDA	☐ OpenMP ☐ DirectX Compute ☐ ATI Stream

	☐ Other	
What kinds of parallelization/multithreading errors does your product look for?	☐ Deadlocks ☐ Race Conditions ☐ Other	☐ Livelocks ☐ Memory leaks
For what kinds of errors does your product provide a mapping to the causing source code?	☐ For all of the above ☐ Deadlocks ☐ Race Conditions	☐ Livelocks ☐ Memory leaks
Does your product use hardware performance counters during analysis?	No	
Does your product generate software traces for later analysis?	No	
In what format are the traces stored?	☐ In a proprietory format ☐ NOPE format	☐ Open Trace Format ☐ Other
Can your product analyse existing software traces?	No	
What trace formats does your product support?	☐ Open Trace Format ☐ Other	☐ NOPE format

Implementation

What programming concepts are fundamental to your product?	☐ Threads (native threads, green threads, n:m threads, etc.) ☐ Message passing ☒ Dataflow concepts ☒ Functional programming concepts ☐ Asynchronous computation with callbacks ☐ Lock-free data structures ☐ Parallel patterns (e.g. Map/Reduce) ☐ Other/Further concepts: ☐ Can't answer
Which memory models form the basis of your product?	☒ Shared memory ☒ Distributed memory ☒ Non-uniform memory access (NUMA)
What special knowledge does a developer/programmer have to have in order to use your product?	☐ Thread programming ☒ Functional programming ☐ Dataflow programming ☐ Parallel patterns ☐ Other
How does a developer model/express parallelism with your product?	☐ Via threads ☒ Via library calls ☐ Via callbacks (filling out a framework) ☐ Via actors (process, agents, etc.) and message passing ☐ Via following a parallel pattern ☐ Other

MWARE
multicore software

How can the correctness of the parallel implementation be reviewed?	☒ Debugging using standard tools ☒ Product offers additional, more suitable tools
What other tools do you suggest for reviewing a parallel implementation that uses your product?	
What programming languages does your product support?	☐ All, product is language-independent ☐ Product is a programming language ☒ C/C++ ☐ Java ☐ C# ☐ Fortran ☐ Other
What features does your product offer for preventing/minimizing common errors in parallel programming?	☐ Deadlock detection ☐ Race condition detection ☐ Transactional memory ☐ Lock-free data structures ☐ Functional semantics ☐ Dataflow semantics ☐ Other
What features of your product make it particularly suitable for the efficient programming of multicore processors?	Offload makes it easy for all types of programmers to quickly write, maintain and debug SPU code. One of the main features of Offload is Call Graph Duplication - a technology which automatically duplicates functions to handle the different combinations of data input from local and shared memory sources.
What features help developers get started using your product?	☒ Known/Familiar programming language ☐ Known abstractions

	☒ Provision of infrastructure (parallel runtime, parallel memory model, etc.) ☒ Plugins for development environments ☐ Specialized development environments ☐ Other
What products do you recommend for the analysis (profiling, debugging, etc.) of software that was implemented using your product?	
What documentation exists for your product?	☒ Meaningful API documentation ☒ Beginner tutorials ☒ Usage examples ☐ Other
What kinds of support do you offer for your product?	☒ On-site support ☐ Phone support ☒ Email support ☐ Conferences ☒ Active community (website, wiki, mailing list, newsgroup, etc.) ☐ Seminars/Workshops ☐ Other

5.11 PGI Suite, The Portland Group

Company Information

Vendor	The Portland Group
Established in	1989

Headquarters	Two Centerpointe, Ste. 320 Lake Oswego, OR 97035 USA
Telephone	503.682.2806
Fax	503.682.2637
Website	www.pgroup.com
Summary	The Portland Group® (a.k.a. PGI®) is a premier supplier of software compilers and tools for parallel computing. The Portland Group offers high performance scalar and parallel Fortran, C and C++ compilers and tools for workstations, servers and clusters based on 64-bit x86 (x64) processors from Intel (Intel 64) and AMD (AMD64 and NVIDIA CUDA-enabled GPGPUs running Linux, MacOS and Windows operating systems.The focus of The Portland Group is to provide the highest performance, production quality compilers and software development tools to the High Performance Technical Computing (HPTC) market. The Portland Group is a wholly-owned subsidiary of STMicroelectronics.

German Service Partner	SMB Kortumstraße 41 448787 Bochum Germany
Telephone	+49 234 5169920
Fax	+49 234 51699229
Website	www.smb-net.de

General Product Information

Product name	PGI Acclerator Workstation/Server PGI CDK Cluster Development Kit PGI Visual Fortran
Market entry	1989
Version	2010
Description	High-performance parallel Fortran, C and C++ compilers and development tools for x64, x86 and GPU enable workstations, servers and clusters running 64-bit or 32-bit Linux, MacOS or Windows. PGI products are a complete self-contained OpenMP/MPI/Accelerator development package.

Product categories	☒ Profiling	☒ Debugging
	☒ Tuning	☐ CASE / Software Modeling

	☐ Algorithm	☒ Runtime
	☐ Framework	☒ Compiler
	☐ Library	☒ Programming language
	☐ Other	
Product markets	☐ Embedded software	☐ Networking software
	☒ Numerical simulations	☐ Office applications
	☒ Technical applications	☒ High Performance Computing
	☐ Computer Games	☐ Other

Customer references	– ANSYS: ANSYS 9.0 – CD-adapco: STAR-CD – ESI Group: PAM-CRASH and PAM-STAMP – Gaussian, Inc.: Gaussian – LSTC: LS-DYNA

Software Development

Software development process model	☒ All	
	☐ V model	☐ Waterfall
	☐ Agile:	☐ Other
Recommended stages		
Other related	– PGDBG parallel graphic debugger	

products	– PGPROF parallel graphic performance profiler
Related products in combination	

Functionality

Supported installation architectures	☒ x86	☒ x86_64
	☐ ia64	☐ ARM
	☐ Sparc	☐ PowerPC
	☐ Cell Broadband Engine	☒ GPUs: NVIDIA CUDA-enabled
	☐ Other	
Supported operating systems	☒ Windows: XP, Vista, 7	
	☒ Linux: Debian, Fedora, Mandriva, RedHat, SUSE	
	☐ AIX	☐ HP-UX
	☐ BSD: FreeBSD	☐ Solaris
	☒ Mac OS X	☐ Other

Application type	☒ Graphical application	☒ Commandline application
	☐ Plugin: Visual Studio	☐ Library/Framework
	☐ Other	

Trial version	Yes
Time limit	15 days
Limitations	Executable files built stop working after the trial period.

Analysis

Supported analysis architectures	☒ All the architectures the product runs on	
	☐ x86	☐ x86_64
	☐ ARM	☐ Sparc
	☐ PowerPC	☐ Cell Broadband Engine
	☐ GPUs	☐ Other
Supported languages	☒ C	☒ C++
	☒ Fortran	☐ Java
	☐ C#	☐ Python
	☐ .NET	☐ OpenCL
	☐ DirectX Compute	☐ ATI Stream
	☐ Other	

Analysis level	☐ On the model level (e.g. UML verification, process checking, sequence diagram analysis)

	☒ On the source code level (e.g. static analysis) ☒ On the communication level (e.g. between processes/threads in MPI) ☒ On the instruction level (e.g. runtime analysis, profiling) ☐ Other/Further levels

Analysis preparation	☐ Restart in analysis environment ☐ Compilation with special parameters: ☐ Manual modifications: ☐ Adding compiler directives ☐ Adding library calls ☐ Other

Analysis results are stored	Yes

Presentation of analysis results	☒ Textual summary ☒ Graphical presentation ☐ Other

Required analysis practice	☐ ☒ ☐ ☐ ☐ Very little A lot

Special knowledge required for analysis	N/A

Analysis results mapping to code	No
Automated advice	Yes
Availability of results	☐ During the analysis ☐ After stopping the analysis ☒ After stopping the analysis and preparing the results
Comparison of analysis runs	Yes
Processing of analysis results is possible with other tools	No

Analysis overhead	2%
Analysis limit	Yes: 64 threads per process, 256 processes
Presentation limit	Yes: same

MWARE multicore software

Profiling

Profiling type	☒ System-wide analysis (like DTrace or oprofile) ☒ Software-specific analysis (like gprof) ☒ Library-specific analysis (like TAU for MPI) ☒ MPI ☒ OpenMP ☐ Intel Threading Building Blocks ☐ OpenCL ☐ DirectX Compute ☐ CUDA ☐ ATI Stream ☒ Other ☐ Other
Profiling method	☒ Statistical sampling ☒ Instrumentation ☐ Simulation ☒ Event-based ☐ Other
Gathered profiling information	☒ Combined runtimes (flat profile) ☐ Runtimes with callgraph ☐ Runtimes with callgraph and trace

Usage of hardware performance counters	Yes
Multicore analysis	☒ CPU load per thread/process ☒ Memory profiling ☐ I/O profiling ☒ Cache profiling ☐ Other
Analysis of processor idle time	No
Measured times	☐ Waiting for I/O ☐ Waiting for thread synchronization ☐ Waiting for acquiring locks ☐ Other

Debugging

What kinds of analysis does your product provide?	☐ Static analysis of source code ☐ Runtime analysis ☒ Runtime analysis with special support for certain libraries ☐ Other

MWARE
multicore software

How does your product perform runtime analysis?	☐ Instrumentation	☐ Event-based
	☐ Simulation	☐ Other
Which libraries does your product support?	☒ MPI	☒ OpenMP
	☐ Intel Threading Building Blocks	
	☐ OpenCL	☐ DirectX Compute
	☐ CUDA	☐ ATI Stream
	☒ Other: PGI Accelerator model	
What kinds of parallelization/multi-threading errors does your product look for?	☐ Deadlocks	☐ Livelocks
	☐ Race Conditions	☐ Memory leaks
	☐ Other	
For what kinds of errors does your product provide a mapping to the causing source code?	☐ For all of the above	
	☐ Deadlocks	☐ Livelocks
	☐ Race Conditions	☐ Memory leaks
Does your product use hardware performance counters during analysis?	Yes	
Does your product generate software traces for later analysis?	No	

In what format are the traces stored?	☐ In a proprietory format	☐ Open Trace Format
	☐ NOPE format	☐ Other
Can your product analyse existing software traces?	No	
What trace formats does your product support?	☐ Open Trace Format	☐ NOPE format
	☐ Other	

Tuning

What kinds of characteristics does your product analyze for performance evaluation?	☒ Processor/Core load	☒ Memory access
	☒ Cache access	☒ Thread communication
	☐ False Sharing	☐ Other
What kinds of techniques does your product use for analysis?	☒ Statistical sampling	☒ Instrumentation
	☐ Simulation	☒ Event-based analysis
	☐ Other	
Does your product use hardware performance counters during analysis?	Yes	
Does your product provide special analysis for parallelization libraries?	Yes	

Which libraries does your product support?	☒ MPI ☒ OpenMP ☐ Intel Threading Building Blocks ☐ OpenCL ☐ DirectX Compute ☐ CUDA ☐ ATI Stream ☒ Other: PGI Accelerator model

Implementation

What programming concepts are fundamental to your product?	☒ Threads (native threads, green threads, n:m threads, etc.) ☒ Message passing ☐ Dataflow concepts ☐ Functional programming concepts ☐ Asynchronous computation with callbacks ☐ Lock-free data structures ☒ Parallel patterns (e.g. Map/Reduce) ☐ Other/Further concepts: ☐ Can't answer
Which memory models form the basis of your product?	☒ Shared memory ☒ Distributed memory ☒ Non-uniform memory access (NUMA)
What special knowledge does a developer/programmer have to have in order	☒ Thread programming ☐ Functional programming ☐ Dataflow programming ☐ Parallel patterns

to use your product?	☐ Other
How does a developer model/express parallelism with your product?	☒ Via threads ☐ Via library calls ☐ Via callbacks (filling out a framework) ☒ Via actors (process, agents, etc.) and message passing ☐ Via following a parallel pattern ☐ Other
How can the correctness of the parallel implementation be reviewed?	☒ Debugging using standard tools ☒ Product offers additional, more suitable tools
What other tools do you suggest for reviewing a parallel implementation that uses your product?	PGDBG parallel debugger
What programming languages does your product support?	☐ All, product is language-independent ☒ Product is a programming language ☒ C/C++ ☐ Java ☐ C# ☒ Fortran ☐ Other
What features does your product offer for preventing/minimizing common errors in parallel programming?	☐ Deadlock detection ☐ Race condition detection ☐ Transactional memory ☐ Lock-free data structures ☐ Functional semantics ☐ Dataflow semantics

	☒ Other: language compliance, runtime error checking
What features of your product make it particularly suitable for the efficient programming of multicore processors?	– State of the art local, global and interprocedural optimizations- Automatic vectorization and SIMD/SSE code generation – Support of OpenMP 3.0 standard – Automatic loop parallelization – Profile-guided optimization – PGI Unified Binary technology to target different 'flavors' of same architecture or heterogeneous architectures – Graphical tools to Debug/Profile Multithreaded/Multiprocess hybrid applications
What features help developers get started using your product?	☒ Known/Familiar programming language ☒ Known abstractions ☒ Provision of infrastructure (parallel runtime, parallel memory model, etc.) ☒ Plugins for development environments ☐ Specialized development environments ☐ Other
What products do you recommend for the analysis (profiling, debugging, etc.) of software that was implemented using your product?	– PGI tools – Allinea TotalView
What documentation exists for your product?	☒ Meaningful API documentation ☒ Beginner tutorials ☒ Usage examples ☒ Other: Language referenceCompiler and Tools manuals

What kinds of support do you offer for your product?	☒ On-site support	☒ Phone support
	☒ Email support	☒ Conferences
	☒ Active community (website, wiki, mailing list, newsgroup, etc.)	
	☒ Seminars/Workshops	☒ Other: Webinars, Tutorials

5.12 Poly-Platform, PolyCore Software

Company Information

Vendor	PolyCore Software, Inc.
Established in	2003

Headquarters	533 Airport Blvd., Suite 400 Burlingame, CA 94010 USA
Telephone	+16505705942
Fax	
Website	www.PolyCoreSoftware.com
Summary	PolyCore Software was founded in 2003 to address challenges brought on by the emergence of multicore chips. The PolyCore Software team brings substantial embedded software and multi-processing experience, and PolyCore Software is a founding board member of the Multicore Association, (http://www.multicore-association.org) and active participant in its standardization working groups. PolyCore Software provides run-time solutions and tools for multicore platforms, serving markets as digital consumer, communication infrastructure, industrial automation, medical and aerospace and defense.

German Service Partner	No

MWARE multicore software

General Product Information

Product name	Poly-Platform which includes Poly-Inspector, Poly-Mapper, Poly-Generator and Poly-Messenger/MCAPI
Market entry	Poly-Messenger was introduced in 2004
Version	N/A
Description	Poly-Platform is multicore communications development tools suite and runtime software which simplifies applications migration to and across multiple compute engines.

Product categories	☒ Profiling	☐ Debugging
	☒ Tuning	☒ CASE / Software Modeling
	☐ Algorithm	☒ Runtime
	☒ Framework	☐ Compiler
	☐ Library	☐ Programming language
	☐ Other	
Product markets	☒ Embedded software	☒ Networking software
	☒ Numerical simulations	☐ Office applications
	☒ Technical applications	☒ High Performance Computing
	☐ Computer Games	☐ Other

Customer references	

Software Development

Software development process model	☒ All	
	☐ V model	☐ Waterfall
	☐ Agile:	☐ Other
Recommended stages	Architecture, design, implmentation, validation, verification	
Other related products	No	

Functionality

Supported installation architectures	☒ x86	☐ x86_64
	☐ ia64	☒ ARM
	☐ Sparc	☒ PowerPC
	☐ Cell Broadband Engine	☐ GPUs:
	☒ Other: BF561 and C55	
Supported operating systems	☒ Windows: XP, Vista, 7	
	☐ Linux: Debian, Fedora, Mandriva, RedHat, SUSE	

	☐ AIX	☐ HP-UX
	☐ BSD: FreeBSD	☐ Solaris
	☐ Mac OS X	☒ Other: Ubuntu Linux

Application type	☐ Graphical application	☐ Commandline application
	☐ Plugin: Visual Studio	☒ Library/Framework
	☐ Other	

Trial version	Yes
Time limit	30 days
Limitations	Number of multicore nodes that are usable

Analysis

Supported analysis architectures	☒ All the architectures the product runs on	
	☐ x86	☐ x86_64
	☐ ARM	☐ Sparc
	☐ PowerPC	☐ Cell Broadband Engine
	☐ GPUs	☐ Other

Supported languages	☒ C	☐ C++
	☐ Fortran	☐ Java
	☐ C#	☐ Python
	☐ .NET	☐ OpenCL
	☐ DirectX Compute	☐ ATI Stream
	☐ Other	

Analysis level	☐ On the model level (e.g. UML verification, process checking, sequence diagram analysis)
	☐ On the source code level (e.g. static analysis)
	☐ On the communication level (e.g. between processes/threads in MPI)
	☒ On the instruction level (e.g. runtime analysis, profiling)
	☐ Other/Further levels

Analysis preparation	☐ Restart in analysis environment
	☒ Compilation with special parameters: gprof flags
	☐ Manual modifications:
	☐ Adding compiler directives ☐ Adding library calls ☐ Other

Analysis results are stored	Yes

Presentation of analysis results	☐ Textual summary ☒ Graphical presentation ☐ Other

Required analysis practice	☒ ☐ ☐ ☐ ☐ Very little A lot
Special knowledge required for analysis	No

Analysis results mapping to code	No
Automated advice	No
Availability of results	☐ During the analysis ☐ After stopping the analysis ☒ After stopping the analysis and preparing the results
Comparison of analysis runs	No
Processing of analysis results is possible with other tools	Yes: any product supporting gprof output

Analysis overhead	N/A
Analysis limit	No
Presentation limit	No

Profiling

Profiling type	☐ System-wide analysis (like DTrace or oprofile) ☒ Software-specific analysis (like gprof) ☐ Library-specific analysis (like TAU for MPI) ☐ MPI ☐ OpenMP ☐ Intel Threading Building Blocks ☐ OpenCL ☐ DirectX Compute ☐ CUDA ☐ ATI Stream ☐ Other ☐ Other
Profiling method	☐ Statistical sampling ☒ Instrumentation ☐ Simulation

	☐ Event-based ☐ Other
Gathered profiling information	☒ Combined runtimes (flat profile) ☒ Runtimes with callgraph ☐ Runtimes with callgraph and trace
Usage of hardware performance counters	No
Multicore analysis	☐ CPU load per thread/process ☐ Memory profiling ☐ I/O profiling ☐ Cache profiling ☐ Other
Analysis of processor idle time	No
Measured times	☐ Waiting for I/O ☐ Waiting for thread synchronization ☐ Waiting for acquiring locks ☐ Other

Tuning

What kinds of characteristics does your product analyze for performance evaluation?	☐ Processor/Core load	☐ Memory access
	☐ Cache access	☐ Thread communication
	☐ False Sharing	☐ Other
What kinds of techniques does your product use for analysis?	☐ Statistical sampling	☒ Instrumentation
	☐ Simulation	☐ Event-based analysis
	☐ Other	
Does your product use hardware performance counters during analysis?	No	
Does your product provide special analysis for parallelization libraries?	No	
Which libraries does your product support?	☐ MPI	☐ OpenMP
	☐ Intel Threading Building Blocks	
	☐ OpenCL	☐ DirectX Compute
	☐ CUDA	☐ ATI Stream
	☐ Other	

MWARE multicore software

Implementation

What programming concepts are fundamental to your product?	☒ Threads (native threads, green threads, n:m threads, etc.) ☒ Message passing ☐ Dataflow concepts ☒ Functional programming concepts ☒ Asynchronous computation with callbacks ☐ Lock-free data structures ☐ Parallel patterns (e.g. Map/Reduce) ☐ Other/Further concepts: ☐ Can't answer
Which memory models form the basis of your product?	☒ Shared memory ☒ Distributed memory ☒ Non-uniform memory access (NUMA)
What special knowledge does a developer/programmer have to have in order to use your product?	☐ Thread programming ☒ Functional programming ☐ Dataflow programming ☐ Parallel patterns ☐ Other
How does a developer model/express parallelism with your product?	☐ Via threads ☒ Via library calls ☐ Via callbacks (filling out a framework) ☒ Via actors (process, agents, etc.) and message passing ☐ Via following a parallel pattern ☐ Other

How can the correctness of the parallel implementation be reviewed?	☒ Debugging using standard tools ☒ Product offers additional, more suitable tools
What other tools do you suggest for reviewing a parallel implementation that uses your product?	Poly-Mapper validates the communications Topology
What programming languages does your product support?	☐ All, product is language-independent ☐ Product is a programming language ☒ C/C++ ☐ Java ☐ C# ☐ Fortran ☐ Other
What features does your product offer for preventing/minimizing common errors in parallel programming?	☐ Deadlock detection ☐ Race condition detection ☐ Transactional memory ☐ Lock-free data structures ☐ Functional semantics ☐ Dataflow semantics ☒ Other: Thread safe message passing
What features of your product make it particularly suitable for the efficient programming of multicore processors?	– Use MCAPI standardized multicore communications API – Tools facilitate the creation of model based applications' communications topology – Generate program elements for the applicaition's multicore communications – Tools enable the architect and developer to quickly move the application across topologies and facilitate the experimentation of alternate configuration and scenarios.

MWARE multicore software

What features help developers get started using your product?	☒ Known/Familiar programming language ☒ Known abstractions ☐ Provision of infrastructure (parallel runtime, parallel memory model, etc.) ☒ Plugins for development environments ☒ Specialized development environments ☐ Other
What products do you recommend for the analysis (profiling, debugging, etc.) of software that was implemented using your product?	
What documentation exists for your product?	☒ Meaningful API documentation ☒ Beginner tutorials ☒ Usage examples ☒ Other: Reference Manuals
What kinds of support do you offer for your product?	☒ On-site support ☒ Phone support ☒ Email support ☒ Conferences ☒ Active community (website, wiki, mailing list, newsgroup, etc.) ☒ Seminars/Workshops ☐ Other

5.13 Prism, CriticalBlue

Company Information

Vendor	CriticalBlue
Established in	2002

Headquarters	181 The Pleasance Edinburgh EH8 9RU UK
Telephone	+441316551500
Fax	+441316551501
Website	www.criticalblue.com
Summary	CriticalBlue is a pioneer of flexible, automated system design solutions that meet the increasing performance, power, and cost demands associated with the delivery of advanced electronic products within today's demanding design schedules. The increasing use of complex, multicore processor architectures has accelerated demand for CriticalBlue's technology and expertise throughout all electronic industry sectors. Headquartered in Edinburgh, Scotland, with offices in San Jose, California, and Tokyo, Japan, the company has delivered multiple solutions for key aspects of embedded software design, including Prism, a multicore embedded software design environment, and Cascade, a software accelerator synthesis technology. The company is funded by European, US Silicon Valley, Japanese venture capitalists and corporate investors. To learn more, please visit www.criticalblue.com.

German Service Partner	No

General Product Information

Product name	Prism
Market entry	2009
Version	2.0
Description	Prism is an award winning Eclipse-based embedded multicore programming system which allows software engineers to easily assess and realize the full potential of multicore processors without significant changes to their development flow. Prism analyzes the behavior of code running on hardware development boards, virtual machines or simulators. It allows engineers to take their existing sequential code, and before making any changes, explore and analyze opportunities for concurrency. Having identified the optimal parallelization strategies in this way, developers will implement parallel structures, and use Prism again to verify efficient and thread-safe operations.

Product categories	☒ Profiling	☐ Debugging
	☒ Tuning	☐ CASE / Software Modeling
	☒ Algorithm	☐ Runtime
	☒ Framework	☐ Compiler
	☐ Library	☐ Programming language
	☐ Other	

Product markets	☒ Embedded software	☒ Networking software
	☒ Numerical simulations	☒ Office applications
	☒ Technical applications	☐ High Performance Computing
	☒ Computer Games	☐ Other

Customer references	

Software Development

Software development process model	☒ All	
	☐ V model	☐ Waterfall
	☐ Agile:	☐ Other
Recommended stages	Prism can be used on purely sequential software that all already exists without changes. The toll will help to guide the user through the process or assesing the value of different parallelization strategies so that the correct one can be chosen before any code changes are committed. Prism can also be used after the code has been parallelized in order to verify that the code will function correctly under all scheduling conditions.	
Other related products	No	

Functionality

Supported installation architectures	☒ x86	☒ x86_64
	☐ ia64	☒ ARM
	☐ Sparc	☒ PowerPC
	☐ Cell Broadband Engine	☐ GPUs:
	☒ Other: MIPS32, Reneses SH4, Cavium cn, MIPS, NEC V850	
Supported operating systems	☒ Windows: XP, Vista, 7	
	☒ Linux: Debian, Fedora, Mandriva, RedHat, SUSE	
	☐ AIX	☐ HP-UX
	☐ BSD: FreeBSD	☐ Solaris
	☐ Mac OS X	☐ Other

Application type	☒ Graphical application	☒ Commandline application
	☐ Plugin: Visual Studio	☐ Library/Framework
	☐ Other	

Trial version	Yes
Time limit	30 days
Limitations	None, just the time limit above. Users can download it from: http://www.criticalblue.com/prism/eval/

Analysis

Supported analysis architectures	☒ All the architectures the product runs on	
	☒ x86	☒ x86_64
	☒ ARM	☐ Sparc
	☒ PowerPC	☐ Cell Broadband Engine
	☐ GPUs	☒ Other: MIPS32, Renesas SH4, NEC V850, Cavium cn, MIPS
Supported languages	☒ C	☒ C++
	☐ Fortran	☐ Java
	☐ C#	☐ Python
	☐ .NET	☐ OpenCL
	☐ DirectX Compute	☐ ATI Stream
	☐ Other	

Analysis level	☐ On the model level (e.g. UML verification, process checking, sequence diagram analysis)
	☐ On the source code level (e.g. static analysis)
	☐ On the communication level (e.g. between processes/threads in MPI)
	☒ On the instruction level (e.g. runtime analysis, profiling)
	☐ Other/Further levels

Analysis preparation	☐ Restart in analysis environment ☐ Compilation with special parameters: ☐ Manual modifications: ☐ Adding compiler directives ☐ Adding library calls ☐ Other

Analysis results are stored	Yes

Presentation of analysis results	☐ Textual summary ☒ Graphical presentation ☐ Other

Required analysis practice	☒ ☐ ☐ ☐ ☐ Very little A lot
Special knowledge required for analysis	No

Analysis results mapping to code	Yes

Automated advice	No
Availability of results	☐ During the analysis ☒ After stopping the analysis ☐ After stopping the analysis and preparing the results
Comparison of analysis runs	No
Processing of analysis results is possible with other tools	No

Analysis overhead	Prism has two methods to extract the profile data. 1. Linked directly into the simulator in which case the overhead is limited. 2. Dynamic instrumentation of the software while execution and this approach will have an impact on performance.
Analysis limit	No
Presentation limit	No

Profiling

Profiling type	☐ System-wide analysis (like DTrace or oprofile) ☒ Software-specific analysis (like gprof) ☐ Library-specific analysis (like TAU for MPI) ☐ MPI

	☐ OpenMP ☐ Intel Threading Building Blocks ☐ OpenCL ☐ DirectX Compute ☐ CUDA ☐ ATI Stream ☐ Other ☐ Other
Profiling method	☐ Statistical sampling ☒ Instrumentation ☒ Simulation ☐ Event-based ☐ Other
Gathered profiling information	☐ Combined runtimes (flat profile) ☐ Runtimes with callgraph ☒ Runtimes with callgraph and trace
Usage of hardware performance counters	No
Multicore analysis	☒ CPU load per thread/process ☒ Memory profiling

	☐ I/O profiling ☒ Cache profiling ☐ Other
Analysis of processor idle time	Yes
Measured times	☐ Waiting for I/O ☒ Waiting for thread synchronization ☒ Waiting for acquiring locks ☐ Other

Tuning

What kinds of characteristics does your product analyze for performance evaluation?	☒ Processor/Core load ☒ Cache access ☒ False Sharing	☒ Memory access ☒ Thread communication ☐ Other
What kinds of techniques does your product use for analysis?	☐ Statistical sampling ☐ Simulation ☐ Other	☒ Instrumentation ☐ Event-based analysis
Does your product use hardware performance counters during analysis?	No	

Does your product provide special analysis for parallelization libraries?	N/A
Which libraries does your product support?	☐ MPI ☐ OpenMP ☐ Intel Threading Building Blocks ☐ OpenCL ☐ DirectX Compute ☐ CUDA ☐ ATI Stream ☐ Other

Implementation

What programming concepts are fundamental to your product?	☒ Threads (native threads, green threads, n:m threads, etc.) ☐ Message passing ☒ Dataflow concepts ☒ Functional programming concepts ☐ Asynchronous computation with callbacks ☒ Lock-free data structures ☒ Parallel patterns (e.g. Map/Reduce) ☐ Other/Further concepts: ☐ Can't answer
Which memory models form the basis of your product?	☒ Shared memory ☐ Distributed memory ☐ Non-uniform memory access (NUMA)

What special knowledge does a developer/programmer have to have in order to use your product?	☐ Thread programming ☐ Functional programming ☐ Dataflow programming ☐ Parallel patterns ☐ Other
How does a developer model/express parallelism with your product?	☒ Via threads ☐ Via library calls ☐ Via callbacks (filling out a framework) ☐ Via actors (process, agents, etc.) and message passing ☐ Via following a parallel pattern ☐ Other
How can the correctness of the parallel implementation be reviewed?	☐ Debugging using standard tools ☒ Product offers additional, more suitable tools
What other tools do you suggest for reviewing a parallel implementation that uses your product?	Prism has a set of verification features to help the programmer ensure that the parallel code is safe.
What programming languages does your product support?	☐ All, product is language-independent ☐ Product is a programming language ☒ C/C++ ☐ Java ☐ C# ☐ Fortran ☐ Other

What features does your product offer for preventing/minimizing common errors in parallel programming?	☐ Deadlock detection ☒ Race condition detection ☐ Transactional memory ☐ Lock-free data structures ☐ Functional semantics ☐ Dataflow semantics ☐ Other
What features of your product make it particularly suitable for the efficient programming of multicore processors?	Prism contains the concept of »what-if« analysis and this is used to explore the amount of natural parallelism in the existing unmodified software. This is achieved by enabling the user to set imaginary partition points on a multicore platform with an imaginary number of cores, and then analyzing the impact of these settings on the overall execution schedule. This is a unique approach and enables the end user to experiment with different platforms and different parallelization techniques - based on regular unmodified code.
What features help developers get started using your product?	☒ Known/Familiar programming language ☒ Known abstractions ☐ Provision of infrastructure (parallel runtime, parallel memory model, etc.) ☒ Plugins for development environments ☒ Specialized development environments ☐ Other
What products do you recommend for the analysis (profiling, debugging, etc.) of software that was implemented using your product?	Software developers can continue using their own IDE just as they have done up to this point. Compiler, profiler and debugger will be the same as before. Once the software has been partitioned and multithreaded, our Prism can be used once again to check for data races. This is a very important verification step but, like other processes inside Prism, it does not require any special language constructs or coding styles.

What documentation exists for your product?	☒ Meaningful API documentation ☒ Beginner tutorials ☒ Usage examples ☐ Other
What kinds of support do you offer for your product?	☒ On-site support ☒ Phone support ☒ Email support ☒ Conferences ☒ Active community (website, wiki, mailing list, newsgroup, etc.) ☒ Seminars/Workshops ☐ Other

5.14 PVS-Studio, Program Verification Systems Co. Ltd

Company Information

Vendor	Program Verification Systems Co Ltd
Established in	2008

Headquarters	Kutuzova 300027, Tula Russia
Telephone	+7 (4872) 38-59-95
Fax	
Website	http://www.viva64.com
Summary	The Company is engaged in software development in the sphere of programs source code analysis. The main spheres of the work are: development and promotion of static code analyzer, PVS-Studio; creation of tools for software testing and quality control. The key technical competences are: 64-bit and parallel programming, and code analysis techniques.

German Service Partner	No

MWARE multicore software

General Product Information

Product name	PVS-Studio
Market entry	2009
Version	3.45
Description	PVS-Studio is a toolset of static source code analyzers designed for detecting 64-bit and parallel errors in software. The tool is designed for developers of state-of-the-art resource-intensive applications. PVS-Studio is integrated in Visual Studio 2005/2008 environment and supports functioning in C/C++ languages. PVS-Studio comprises Viva64 analyzer for detecting errors in 64-bit programs and VivaMP analyzer for detecting errors in parallel programs built with the help of OpenMP technology. Estimation of time and complexity of code migration process to 64-bit systems is an important feature of the tool.

Product categories	☐ Profiling	☐ Debugging
	☐ Tuning	☐ CASE / Software Modeling
	☐ Algorithm	☐ Runtime
	☐ Framework	☐ Compiler
	☐ Library	☐ Programming language
	☒ Other: Static analyzer of source code	
Product markets	☐ Embedded software	☐ Networking software
	☒ Numerical simulations	☒ Office applications
	☒ Technical applications	☒ High Performance Computing

	☒ Computer Games	☐ Other

Customer references	– E.G.S.: Leios Components – Loki Library

Software Development

Software development process model	☒ All	
	☐ V model	☐ Waterfall
	☐ Agile:	☐ Other
Recommended stages	Construction stage	
Other related products	No	

Functionality

Supported installation architectures	☒ x86	☒ x86_64
	☐ ia64	☐ ARM
	☐ Sparc	☐ PowerPC
	☐ Cell Broadband Engine	☐ GPUs:
	☐ Other	

Supported operating systems	☒ Windows: XP, Vista, 7 ☐ Linux: Debian, Fedora, Mandriva, RedHat, SUSE ☐ AIX ☐ HP-UX ☐ BSD: FreeBSD ☐ Solaris ☐ Mac OS X ☐ Other

Application type	☐ Graphical application ☐ Commandline application ☐ Plugin: Visual Studio ☐ Library/Framework ☐ Other

Trial version	Yes
Time limit	30 days
Limitations	PVS-Studio demo-version shows location only of some of the defects and errors in the code (although it detects them all). But for PortSample and ParallelSample we have made an exception - for them all the defects are shown.

Analysis

Supported analysis architectures	☐ All the architectures the product runs on ☐ x86 ☐ x86_64 ☐ ARM ☐ Sparc

	☐ PowerPC	☐ Cell Broadband Engine
	☐ GPUs	☐ Other
Supported languages	☐ C	☐ C++
	☐ Fortran	☐ Java
	☐ C#	☐ Python
	☐ .NET	☐ OpenCL
	☐ DirectX Compute	☐ ATI Stream
	☐ Other	

Analysis level	☐ On the model level (e.g. UML verification, process checking, sequence diagram analysis)
	☐ On the source code level (e.g. static analysis)
	☐ On the communication level (e.g. between processes/threads in MPI)
	☐ On the instruction level (e.g. runtime analysis, profiling)
	☐ Other/Further levels

Analysis preparation	☐ Restart in analysis environment
	☐ Compilation with special parameters:
	☐ Manual modifications:
	☐ Adding compiler directives ☐ Adding library calls ☐ Other

Analysis results are stored	No

Presentation of analysis results	☐ Textual summary ☐ Graphical presentation ☐ Other

Required analysis practice	N/A
Special knowledge required for analysis	No

Analysis results mapping to code	No
Automated advice	No
Availability of results	☐ During the analysis ☐ After stopping the analysis ☐ After stopping the analysis and preparing the results
Comparison of analysis runs	No

Processing of analysis results is possible with other tools	No

Analysis overhead	N/A
Analysis limit	N/A
Presentation limit	N/A

Debugging

What kinds of analysis does your product provide?	☐ Static analysis of source code ☐ Runtime analysis ☐ Runtime analysis with special support for certain libraries ☐ Other	
How does your product perform runtime analysis?	☐ Instrumentation ☐ Simulation	☐ Event-based ☐ Other
Which libraries does your product support?	☐ MPI ☐ Intel Threading Building Blocks ☐ OpenCL	☐ OpenMP ☐ DirectX Compute

	☐ CUDA	☐ ATI Stream
	☐ Other	
What kinds of parallelization/multithreading errors does your product look for?	☐ Deadlocks	☐ Livelocks
	☐ Race Conditions	☐ Memory leaks
	☐ Other	
For what kinds of errors does your product provide a mapping to the causing source code?	☐ For all of the above	
	☐ Deadlocks	☐ Livelocks
	☐ Race Conditions	☐ Memory leaks
Does your product use hardware performance counters during analysis?	N/A	
Does your product generate software traces for later analysis?	N/A	
Can your product analyse existing software traces?	No	

5.15 SAMG, Fraunhofer SCAI

Company Information

Vendor	Fraunhofer SCAI
Established in	1950

Headquarters	Schloss Birlinghoven 53757 Sankt Augustin Germany
Telephone	+49 (2241) 14-2759
Fax	
Website	www.scai.fraunhofer.de
Summary	www.fraunhofer.de www.scai.fraunhofer.de

German Service Partner	N/A

MWARE multicore software

General Product Information

Product name	SAMG - Algebraic Multigrid for Systems
Market entry	2000
Version	Release 24a2
Description	Fast Linear Solver

Product categories	☐ Profiling	☐ Debugging
	☐ Tuning	☐ CASE / Software Modeling
	☒ Algorithm	☐ Runtime
	☒ Framework	☐ Compiler
	☒ Library	☐ Programming language
	☐ Other	
Product markets	☐ Embedded software	☐ Networking software
	☒ Numerical simulations	☐ Office applications
	☒ Technical applications	☒ High Performance Computing
	☐ Computer Games	☐ Other

Customer references	

Software Development

Software development process model	☐ All	
	☐ V model	☐ Waterfall
	☐ Agile:	☐ Other
Recommended stages		
Other related products	No	

Functionality

Supported installation architectures	☒ x86	☒ x86_64
	☒ ia64	☒ ARM
	☒ Sparc	☒ PowerPC
	☐ Cell Broadband Engine	☐ GPUs:
	☐ Other	
Supported operating systems	☒ Windows: XP, Vista, 7	
	☒ Linux: Debian, Fedora, Mandriva, RedHat, SUSE	
	☒ AIX	☒ HP-UX
	☒ BSD: FreeBSD	☒ Solaris
	☒ Mac OS X	☐ Other

Application type	☐ Graphical application	☐ Commandline application
	☐ Plugin: Visual Studio	☒ Library/Framework
	☐ Other	

Trial version	Yes
Time limit	30 days
Limitations	none

Implementation

What programming concepts are fundamental to your product?	☒ Threads (native threads, green threads, n:m threads, etc.)
	☒ Message passing
	☐ Dataflow concepts
	☐ Functional programming concepts
	☐ Asynchronous computation with callbacks
	☐ Lock-free data structures
	☐ Parallel patterns (e.g. Map/Reduce)
	☐ Other/Further concepts:
	☐ Can't answer

Which memory models form the basis of your product?	☒ Shared memory ☒ Distributed memory ☒ Non-uniform memory access (NUMA)
What special knowledge does a developer/programmer have to have in order to use your product?	☐ Thread programming ☐ Functional programming ☐ Dataflow programming ☐ Parallel patterns ☒ Other: numerical modeling
How does a developer model/express parallelism with your product?	☐ Via threads ☒ Via library calls ☐ Via callbacks (filling out a framework) ☒ Via actors (process, agents, etc.) and message passing ☐ Via following a parallel pattern ☐ Other
How can the correctness of the parallel implementation be reviewed?	☒ Debugging using standard tools ☐ Product offers additional, more suitable tools
What other tools do you suggest for reviewing a parallel implementation that uses your product?	
What programming languages does your product support?	☐ All, product is language-independent ☐ Product is a programming language ☒ C/C++ ☐ Java

	☐ C#	☒ Fortran
	☐ Other	
What features does your product offer for preventing/minimizing common errors in parallel programming?	☐ Deadlock detection	☐ Race condition detection
	☐ Transactional memory	☐ Lock-free data structures
	☐ Functional semantics	☐ Dataflow semantics
	☐ Other	
What features of your product make it particularly suitable for the efficient programming of multicore processors?	It is usable as (typically the most expensive) parallel component of an overall parallel numerical simulation code	
What features help developers get started using your product?	☐ Known/Familiar programming language	
	☐ Known abstractions	
	☐ Provision of infrastructure (parallel runtime, parallel memory model, etc.)	
	☐ Plugins for development environments	
	☐ Specialized development environments	
	☒ Other: simple library interface;demo programs	
What products do you recommend for the analysis (profiling, debugging, etc.) of software that was implemented using your product?	– MPI tracing tools (such as Vampir, ITAC, ...) – Threading tools	

What documentation exists for your product?	☐ Meaningful API documentation ☒ Beginner tutorials ☒ Usage examples ☐ Other
What kinds of support do you offer for your product?	☒ On-site support ☒ Phone support ☒ Email support ☐ Conferences ☐ Active community (website, wiki, mailing list, newsgroup, etc.) ☒ Seminars/Workshops ☐ Other

5.16 Scalasca, Forschungszentrum Jülich

Company Information

Vendor	Forschungszentrum Jülich
Established in	1956

Headquarters	Jülich Supercomputing Centre 52425 Jülich Germany
Telephone	+49 2461 61-1583
Fax	+49 2461 61-6656
Website	http://www.fz-juelich.de
Summary	The product is jointly developed by Forschungszentrum Jülich GmbH and the German Research School for Simulation Sciences GmbH. Forschungszentrum Jülich GmbH (Year of foundation: 1956) Forschungszentrum Jülich pursues cutting-edge interdisciplinary research on solving the grand challenges facing society in the fields of health, energy and the environment, and also information technologies. In combination with its two key competencies – physics and supercomputing – work at Jülich focuses on both long-term, fundamental and multidisciplinary contributions to science and technology as well as on specific technological applications. With a staff of about 4400, Jülich – a member of the Helmholtz Association – is one of the largest research centers in Europe. Jülich Supercomputing Centre at the Forschungszentrum, where Scalasca is

	developed, provides supercomputer resources, IT tools, methods and knowhow for the Forschungszentrum and for national and European users. It operates the central supercomputers and server systems as well as the campus-wide computer networks and communication systems. In support of its mission, Jülich Supercomputing Centre also conducts research in computational science, computer science, and mathematics. German Research School for Simulation Sciences GmbH (Year of foundation: 2007) The German Research School for Simulation Sciences is a joint graduate school of RWTH Aachen University and Forschungszentrum Jülich. Combining the specific strengths of the two founders in the fields of science, engineering, and high-performance computing in an unprecedented, synergistic manner, the school provides a unique environment for cutting-edge interdisciplinary research and education in the applications and methods of simulation in science and engineering. Equipped with dedicated modern facilities in Aachen and on the Jülich campus, and privileged access to world-class computing and visualization resources, the mission of our school is to offer advanced interdisciplinary graduate training programs including a master\'s as well as a doctoral program. The school\'s Laboratory for Parallel Programming, which participates in the development of Scalasca, specializes in performance-analysis methods for parallel programs and offers courses related to parallel programming.

German Service Partner	N/A

General Product Information

Product name	Scalasca

Market entry	2008
Version	1.3
Description	Scalasca is an open-source toolset that can be used to analyze the performance behavior of parallel applications and to identify opportunities for optimization. Target applications include simulation codes from science and engineering written in C, C++ and Fortran and based on the parallel programming interfaces MPI and/or OpenMP. Scalasca has been specifically designed for use on large-scale systems including IBM Blue Gene and Cray XT, but is also well suited for a wide range of small- and medium-scale HPC platforms. Scalasca integrates runtime summaries, which are suitable to obtain a performance overview, with in-depth studies of concurrent behavior via event tracing. The traces are analyzed to identify wait states that occur, for example, as a result of unevenly distributed workloads. Especially when trying to scale communication-intensive applications to large processor counts, such wait states can present serious challenges to achieving good performance. Scalasca is jointly developed by Forschungszentrum Jülich and the German Research School for Simulation Sciences in Aachen. The software is available for free download under the New BSD license.

Product categories	☒ Profiling	☐ Debugging
	☐ Tuning	☐ CASE / Software Modeling
	☐ Algorithm	☐ Runtime
	☐ Framework	☐ Compiler
	☐ Library	☐ Programming language
	☐ Other	
Product markets	☐ Embedded software	☐ Networking software

	☒ Numerical simulations	☐ Office applications
	☐ Technical applications	☒ High Performance Computing
	☐ Computer Games	☐ Other

Customer references	– Bull (France) – Dassault Aviation (France) – GNS (Germany) – MAGMA (Germany) – RECOM (Germany) – Shell (Netherlands) – Sun Microsystems (USA, Singapore, India) – Qontix (UK) – Large number of supercomputing centres and universities

Software Development

Software development process model	☒ All	
	☐ V model	☐ Waterfall
	☐ Agile:	☐ Other
Recommended stages	Implementation / performance optimization	
Other related products	SIONlib, alibrary to facilitate efficient massively-parallel task-local I/O	
Related products in combination	SIONlib can be used as an optional feature inside Scalasca to accelerate the writing of trace files	

Functionality

Supported installation architectures	☒ x86	☒ x86_64
	☒ ia64	☐ ARM
	☒ Sparc	☒ PowerPC
	☐ Cell Broadband Engine	☐ GPUs:
	☒ Other: MIPS, NEC vector processor	
Supported operating systems	☐ Windows: XP, Vista, 7	
	☒ Linux: Debian, Fedora, Mandriva, RedHat, SUSE	
	☒ AIX	☐ HP-UX
	☐ BSD: FreeBSD	☒ Solaris
	☐ Mac OS X	☒ Other: Cray XT, IBM Blue Gene, NEC SX

Application type	☒ Graphical application	☒ Commandline application
	☐ Plugin: Visual Studio	☒ Library/Framework
	☐ Other	

Trial version	Yes
Time limit	Our software is free and available under the New BSD license. It can be used for an infinite amount of time.

Limitations	None

Analysis

Supported analysis architectures	☒ All the architectures the product runs on	
	☐ x86	☐ x86_64
	☐ ARM	☐ Sparc
	☐ PowerPC	☐ Cell Broadband Engine
	☐ GPUs	☐ Other
Supported languages	☒ C	☒ C++
	☒ Fortran	☐ Java
	☐ C#	☐ Python
	☐ .NET	☐ OpenCL
	☐ DirectX Compute	☐ ATI Stream
	☐ Other	

Analysis level	☐ On the model level (e.g. UML verification, process checking, sequence diagram analysis)
	☐ On the source code level (e.g. static analysis)
	☒ On the communication level (e.g. between processes/threads in MPI)
	☐ On the instruction level (e.g. runtime analysis, profiling)
	☒ Other/Further levels

Analysis preparation	☒ Restart in analysis environment ☒ Compilation with special parameters: The application software must be relinked with our measurement library.Per default, special compiler parameters are used for instrumentation purposes. However, the user has the option to perform the instrumentation manually and not to use the compiler parameters. ☒ Manual modifications: ☐ Adding compiler directives ☐ Adding library calls ☒ Other

Analysis results are stored	Yes

Presentation of analysis results	☒ Textual summary ☒ Graphical presentation ☐ Other

Required analysis practice	☐ ☐ ☒ ☐ ☐ Very little A lot
Special knowledge required for analysis	Yes: Understanding of the performance metrics unique to the tool and understanding of the data model used to present the results.

Analysis results mapping to code	Yes
Automated advice	Yes
Availability of results	☐ During the analysis ☐ After stopping the analysis ☒ After stopping the analysis and preparing the results
Comparison of analysis runs	Yes
Processing of analysis results is possible with other tools	Yes: Jumpshot, Paraver, TAU, Paraprof, Vampir

Analysis overhead	Configuration / application dependent
Analysis limit	Yes: Depends on available processors and memory
Presentation limit	Yes: Depends on screen resolution and available memory

Profiling

Profiling type	☐ System-wide analysis (like DTrace or oprofile) ☒ Software-specific analysis (like gprof) ☒ Library-specific analysis (like TAU for MPI) ☒ MPI ☒ OpenMP ☐ Intel Threading Building Blocks ☐ OpenCL ☐ DirectX Compute ☐ CUDA ☐ ATI Stream ☐ Other ☐ Other
Profiling method	☐ Statistical sampling ☒ Instrumentation ☐ Simulation ☒ Event-based ☐ Other
Gathered profiling information	☐ Combined runtimes (flat profile) ☒ Runtimes with callgraph ☒ Runtimes with callgraph and trace

Usage of hardware performance counters	Yes
Multicore analysis	☒ CPU load per thread/process ☐ Memory profiling ☒ I/O profiling ☒ Cache profiling ☐ Other
Analysis of processor idle time	Yes
Measured times	☐ Waiting for I/O ☒ Waiting for thread synchronization ☐ Waiting for acquiring locks ☒ Other:

5.17 Visual Studio 2010, Microsoft

Company Information

Vendor	Microsoft
Established in	1975

Headquarters	Konrad-Zuse-Str. 1 85716 Unterschleißheim Germany
Telephone	+49 (0) 89 3176 2895
Fax	+49 (0) 89 3176 2830
Website	http://www.microsoft.com/de/de/default.aspx
Summary	Microsoft ist der weltweit führende Hersteller von Standardsoftware, Services und Lösungen, die Menschen und Unternehmen aller Branchen und Größen helfen, ihr Potenzial voll zu entfalten. Sicherheit und Zuverlässigkeit, Innovation und Integration sowie Offenheit und Interoperabilität stehen bei der Entwicklung der Microsoft-Produkte im Mittelpunkt.

German Service Partner	Yes

MWARE multicore software

General Product Information

Product name	Visual Studio 2010
Market entry	2005
Version	10
Description	Visual Studio 2010 supports the overall application lifcecycle management. It has enhanced support for the developement of applications for multicore hardware on the microsoft windows operating systems.

Product categories	☒ Profiling	☒ Debugging
	☒ Tuning	☒ CASE / Software Modeling
	☒ Algorithm	☒ Runtime
	☒ Framework	☒ Compiler
	☒ Library	☒ Programming language
	☐ Other	
Product markets	☒ Embedded software	☒ Networking software
	☒ Numerical simulations	☒ Office applications
	☒ Technical applications	☒ High Performance Computing
	☒ Computer Games	☒ Other: Commercial applications, database applications, web-applications, N-Tier applications, cloud computing

Customer references	

Software Development

Software development process model	☒ All	
	☐ V model	☐ Waterfall
	☐ Agile:	☐ Other
Recommended stages	For all stages	
Other related products	No	

Functionality

Supported installation architectures	☒ x86	☒ x86_64
	☒ ia64	☒ ARM
	☐ Sparc	☐ PowerPC
	☐ Cell Broadband Engine	☐ GPUs:
	☐ Other	
Supported operating systems	☒ Windows: XP, Vista, 7	
	☐ Linux: Debian, Fedora, Mandriva, RedHat, SUSE	

	☐ AIX	☐ HP-UX
	☐ BSD: FreeBSD	☐ Solaris
	☐ Mac OS X	☒ Other: Windows Server Operating Systems (2003 ...)

Application type	☒ Graphical application	☒ Commandline application
	☐ Plugin: Visual Studio	☐ Library/Framework
	☐ Other	

Trial version	Yes
Time limit	3 months
Limitations	None

Analysis

Supported analysis architectures	☐ All the architectures the product runs on	
	☒ x86	☐ x86_64
	☒ ARM	☐ Sparc
	☐ PowerPC	☐ Cell Broadband Engine
	☐ GPUs	☐ Other

Supported languages	☐ C	☒ C++
	☐ Fortran	☐ Java
	☒ C#	☒ Python
	☒ .NET	☐ OpenCL
	☒ DirectX Compute	☐ ATI Stream
	☒ Other: F#, VB.NET	

Analysis level	☒ On the model level (e.g. UML verification, process checking, sequence diagram analysis)
	☒ On the source code level (e.g. static analysis)
	☒ On the communication level (e.g. between processes/threads in MPI)
	☒ On the instruction level (e.g. runtime analysis, profiling)
	☐ Other/Further levels

Analysis preparation	☐ Restart in analysis environment
	☐ Compilation with special parameters:
	☐ Manual modifications:
	☐ Adding compiler directives ☐ Adding library calls ☐ Other

Analysis results are stored	Yes

Presentation of analysis results	☒ Textual summary ☒ Graphical presentation ☐ Other

Required analysis practice	☐ ☒ ☐ ☐ ☐ Very little A lot
Special knowledge required for analysis	No

Analysis results mapping to code	Yes
Automated advice	No
Availability of results	☐ During the analysis ☒ After stopping the analysis ☐ After stopping the analysis and preparing the results
Comparison of analysis runs	Yes
Processing of analysis results is possbile with other tools	No

Analysis overhead	Depends of the details of the analysis
Analysis limit	No
Presentation limit	No

Profiling

Profiling type	☒ System-wide analysis (like DTrace or oprofile) ☒ Software-specific analysis (like gprof) ☒ Library-specific analysis (like TAU for MPI) ☒ MPI ☒ OpenMP ☒ Intel Threading Building Blocks ☐ OpenCL ☐ DirectX Compute ☐ CUDA ☐ ATI Stream ☐ Other: ☐ Other
Profiling method	☒ Statistical sampling ☒ Instrumentation

	☐ Simulation ☒ Event-based ☐ Other
Gathered profiling information	☐ Combined runtimes (flat profile) ☒ Runtimes with callgraph ☐ Runtimes with callgraph and trace
Usage of hardware performance counters	☐ Yes
Multi-Core analysis	☒ CPU load per thread/process ☒ Memory profiling ☒ I/O profiling ☐ Cache profiling ☐ Other
Analysis of processor idle time	Yes
Measured times	☒ Waiting for I/O ☒ Waiting for thread synchronization ☐ Waiting for acquiring locks ☐ Other

Debugging

What kinds of analysis does your product provide?	☒ Static analysis of source code	
	☒ Runtime analysis	
	☐ Runtime analysis with special support for certain libraries	
	☐ Other	
How does your product perform runtime analysis?	☒ Instrumentation	☒ Event-based
	☐ Simulation	☐ Other
Which libraries does your product support?	☐ MPI	☐ OpenMP
	☐ Intel Threading Building Blocks	
	☐ OpenCL	☐ DirectX Compute
	☐ CUDA	☐ ATI Stream
	☐ Other	
What kinds of parallelization/multi-threading errors does your product look for?	☒ Deadlocks	☒ Livelocks
	☒ Race Conditions	☐ Memory leaks
	☐ Other	
For what kinds of errors does your product provide a mapping to the causing source code?	☐ For all of the above	
	☒ Deadlocks	☐ Livelocks
	☐ Race Conditions	☐ Memory leaks

Does your product use hardware performance counters during analysis?	Yes	
Does your product generate software traces for later analysis?	Yes	
In what format are the traces stored?	☒ In a proprietory format	☐ Open Trace Format
	☐ NOPE format	☐ Other
Can your product analyse existing software traces?	Yes	
What trace formats does your product support?	☐ Open Trace Format	☐ NOPE format
	☒ Other: proprietory	

Tuning

What kinds of characteristics does your product analyze for performance evaluation?	☒ Processor/Core load	☒ Memory access
	☐ Cache access	☒ Thread communication
	☐ False Sharing	☐ Other
What kinds of techniques does your product use for analysis?	☒ Statistical sampling	☒ Instrumentation
	☐ Simulation	☒ Event-based analysis
	☐ Other	

Does your product use hardware performance counters during analysis?	Yes
Does your product provide special analysis for parallelization libraries?	Yes
Which libraries does your product support?	☒ MPI ☒ OpenMP ☒ Intel Threading Building Blocks ☐ OpenCL ☐ DirectX Compute ☐ CUDA ☐ ATI Stream ☐ Other

Implementation

What programming concepts are fundamental to your product?	☒ Threads (native threads, green threads, n:m threads, etc.) ☐ Message passing ☐ Dataflow concepts ☒ Functional programming concepts ☒ Asynchronous computation with callbacks ☒ Lock-free data structures ☒ Parallel patterns (e.g. Map/Reduce) ☒ Other/Further concepts: Task Parallel Library ☐ Can't answer

MWARE multicore software

Which memory models form the basis of your product?	☒ Shared memory ☐ Distributed memory ☐ Non-uniform memory access (NUMA)
What special knowledge does a developer/programmer have to have in order to use your product?	☒ Thread programming ☐ Functional programming ☐ Dataflow programming ☐ Parallel patterns ☐ Other
How does a developer model/express parallelism with your product?	☒ Via threads ☐ Via library calls ☐ Via callbacks (filling out a framework) ☐ Via actors (process, agents, etc.) and message passing ☐ Via following a parallel pattern ☒ Other: Tasks
How can the correctness of the parallel implementation be reviewed?	☒ Debugging using standard tools ☒ Product offers additional, more suitable tools
What other tools do you suggest for reviewing a parallel implementation that uses your product?	Parallel Debugger
What programming languages does your product support?	☐ All, product is language-independent ☐ Product is a programming language ☒ C/C++ ☐ Java

	☒ C# ☐ Fortran ☒ Other: VB.NET, F#, Python, Ruby, ...
What features does your product offer for preventing/minimizing common errors in parallel programming?	☐ Deadlock detection ☐ Race condition detection ☒ Transactional memory ☐ Lock-free data structures ☐ Functional semantics ☐ Dataflow semantics ☐ Other
What features of your product make it particularly suitable for the efficient programming of multicore processors?	Visual Profiler, visual debugger and task parallel library.
What features help developers get started using your product?	☒ Known/Familiar programming language ☒ Known abstractions ☐ Provision of infrastructure (parallel runtime, parallel memory model, etc.) ☐ Plugins for development environments ☒ Specialized development environments ☐ Other
What products do you recommend for the analysis (profiling, debugging, etc.) of software that was implemented using your product?	Visual Studio 2010

What documentation exists for your product?	☒ Meaningful API documentation ☒ Beginner tutorials ☒ Usage examples ☐ Other
What kinds of support do you offer for your product?	☒ On-site support ☒ Phone support ☒ Email support ☒ Conferences ☒ Active community (website, wiki, mailing list, newsgroup, etc.) ☒ Seminars/Workshops ☒ Other: Startup programs for partners

5.18 Zoom, RotateRight LLC

Company Information

Vendor	RotateRight LLC
Established in	2006

Headquarters	5524 Woodlawn Ave N 98103 Seattle USA
Telephone	
Fax	
Website	
Summary	Development tools and performance optimization services

German Service Partner	No

MWARE multicore software

General Product Information

Product name	Zoom
Market entry	2007
Version	1.6.3
Description	Software profiler for Linux

Product categories	☒ Profiling	☐ Debugging
	☒ Tuning	☐ CASE / Software Modeling
	☐ Algorithm	☐ Runtime
	☐ Framework	☐ Compiler
	☐ Library	☐ Programming language
	☐ Other	
Product markets	☒ Embedded software	☒ Networking software
	☒ Numerical simulations	☒ Office applications
	☒ Technical applications	☒ High Performance Computing
	☒ Computer Games	☐ Other

Customer references	

Software Development

Software development process model	☒ All ☐ V model ☐ Waterfall ☐ Agile: ☐ Other
Recommended stages	
Other related products	No

Functionality

Supported installation architectures	☒ x86 ☒ x86_64 ☐ ia64 ☒ ARM ☐ Sparc ☒ PowerPC ☒ Cell Broadband Engine ☐ GPUs: ☐ Other
Supported operating systems	☐ Windows: XP, Vista, 7 ☒ Linux: Debian, Fedora, Mandriva, RedHat, SUSE ☐ AIX ☐ HP-UX ☐ BSD: FreeBSD ☐ Solaris ☐ Mac OS X ☐ Other

Application type	☒ Graphical application	☒ Commandline application
	☐ Plugin: Visual Studio	☐ Library/Framework
	☐ Other	

Trial version	Yes
Time limit	30 days
Limitations	None

Analysis

Supported analysis architectures	☐ All the architectures the product runs on	
	☒ x86	☒ x86_64
	☒ ARM	☐ Sparc
	☒ PowerPC	☒ Cell Broadband Engine
	☐ GPUs	☐ Other
Supported languages	☒ C	☒ C++
	☒ Fortran	☐ Java
	☐ C#	☐ Python

	☐ .NET	☐ OpenCL
	☐ DirectX Compute	☐ ATI Stream
	☐ Other	

Analysis level	☐ On the model level (e.g. UML verification, process checking, sequence diagram analysis) ☒ On the source code level (e.g. static analysis) ☐ On the communication level (e.g. between processes/threads in MPI) ☒ On the instruction level (e.g. runtime analysis, profiling) ☐ Other/Further levels

Analysis preparation	☐ Restart in analysis environment ☐ Compilation with special parameters: ☐ Manual modifications: ☐ Adding compiler directives ☐ Adding library calls ☐ Other

Analysis results are stored	Yes

Presentation of analysis results	☒ Textual summary

	☒ Graphical presentation ☐ Other

Required analysis practice	☐ ☒ ☐ ☐ ☐ Very little A lot
Special knowledge required for analysis	No

Analysis results mapping to code	Yes
Automated advice	Yes
Availability of results	☐ During the analysis ☐ After stopping the analysis ☒ After stopping the analysis and preparing the results
Comparison of analysis runs	No
Processing of analysis results with other tools	No

Analysis overhead	2%
Analysis limit	No
Presentation limit	No

Profiling

Profiling type	☒ System-wide analysis (like DTrace or oprofile) ☒ Software-specific analysis (like gprof) ☐ Library-specific analysis (like TAU for MPI) ☐ MPI ☐ OpenMP ☐ Intel Threading Building Blocks ☐ OpenCL ☐ DirectX Compute ☐ CUDA ☐ ATI Stream ☐ Other ☐ Other
Profiling method	☒ Statistical sampling ☐ Instrumentation ☐ Simulation

	☒ Event-based ☐ Other
Gathered profiling information	☐ Combined runtimes (flat profile) ☒ Runtimes with callgraph ☒ Runtimes with callgraph and trace
Usage of hardware performance counters	Yes
Multi-Core analysis	☒ CPU load per thread/process ☐ Memory profiling ☐ I/O profiling ☐ Cache profiling ☐ Other
Analysis of processor idle time	No
Measured times	☐ Waiting for I/O ☐ Waiting for thread synchronization ☐ Waiting for acquiring locks ☐ Other

Tuning

What kinds of characteristics does your product analyze for performance evaluation?	☒ Processor/Core load	☒ Memory access
	☒ Cache access	☐ Thread communication
	☐ False Sharing	☐ Other
What kinds of techniques does your product use for analysis?	☒ Statistical sampling	☐ Instrumentation
	☐ Simulation	☒ Event-based analysis
	☐ Other	
Does your product use hardware performance counters during analysis?	Yes	
Does your product provide special analysis for parallelization libraries?	No	
Which libraries does your product support?	☐ MPI	☐ OpenMP
	☐ Intel Threading Building Blocks	
	☐ OpenCL	☐ DirectX Compute
	☐ CUDA	☐ ATI Stream
	☐ Other	

6 The MWare Project

The MWare project, funded by the Fraunhofer Gesellschaft, is a cooperation of four Fraunhofer institutes: ITWM, IAO, SCAI, IESE. The project goal is to develop processes, methods and tools to support software development for multicore architectures and to provide a market survey of currently available methods and tools.

The main reason for these activities is the increasing spread of multicore processors and the fact that the optimal performance usage of this processor type requires a fundamental shift of methods and software development approaches compared to current development processes for singlecore architectures.

The goal is to actively support this paradigm shift and to provide the software engineers with the required resources for future software development. Therefore, the above-named Fraunhofer institutes combine their competences to optimize software development processes, to develop efficient algorithms, to parallelize applications, to optimize performance, to develop tools and to select and evaluate soft- and hardware or to port application programs to new architectures. The project is also focused on supporting the software development for embedded systems.

www.mware.fraunhofer.de/EN

MWARE multicore software

Vendor View

Performance Debugging For Multithreaded Applications

Acumem ThreadSpotter™

- The first software for analysis of Multithreaded applications
- Find and fix slowspots in your application code
- Identify and eliminate multicore bottlenecks
- Analyze and improve memory bandwidth utilization

Your application can run faster!

Acumem´s performance debugger ThreadSpotter ™ analyses your application code in new ways spotting opportunities for performance tuning that no other tool can find. ThreadSpotter's insightful performance analysis offer very detailed advice on how to enhance the performance of your application.

We show you how!

Current application performance is evaluated across four performance categories:

Memory Bandwidth	Does your application consume the optimal level of memory bandwidth?
Memory Latency	How often will your processors stall waiting for data from memory?
Data Locality	How much data that is never used will your application transport and store?
Thread Communication	Are your threads interacting in a good or a destructive manner?

ThreadSpotter gives developers hands on advice identifying WHAT is causing the problem, HOW to improve each aspect and WHERE in the source code to look. Make sure your applications are prepared for multicore!

Operating systems	Linux 2.6, Solaris 10/x86, Windows
CPU	All x86/x86-64 processors from Intel and AMD
Supported languages	Compiled languages, such as: C, C++, Fortran, Ada.

info@acumem.com

acumem multicore performance

www.acumem.com

Akka Open Source

Core features

- Actors
- STM – Software Transactional Memory
- Transactors – Transactional Actors – atomic composable message flows
- Remote Actors – a high performant implementation, based on scalable NIO and Protobuf
- Agents
- Dispatchers – implementing a lot of different dipatching/scheduling algorithms
- Serialization – supporting numerous different protocols
- Fault tolerance through Erlang-style Supervisor Hierarchies

Add-on modules

- Microkernel – stand-alone application server
- Pluggable Persistence - Cassandra, MongoDB or Redis
- Apache Camel integration
- AMQP integration
- JTA integration
- Spring integration
- Guice Integration
- Scheduling service
- Publish subscribe model - remote

Support and samples

- Extensive reference documentation
- The mailing list: Akka User List
- Public issue tracking
- Semi-frequent QA secured major releases
- Samples
- Articles

Drottninggatan 108, 113 60 Stockholm ++ +46 733 777 123 ++ jonas@jonasboner.com

Akka Enterprise

Enterprise level cluster support

- Two different backends – Peer-to-peer based module or centralized ZooKeeper based module
- Runtime Actor code provisioning and remote class loading
- Clustered Actor registration/lookup service
- Distributed message routing service
- Clustered publish subscribe service
- No limitations on the number if nodes
- Cluster membership API

Provisioning

- OSGi-based provisioning of the system as the load changes
- Add and remove nodes on the fly
- Rolling upgrades of user system and/or Akka modules
- Management through Akka Dashboard

Remote

- SSL based communication for Remote Actors
- WebSocket service for Remote Actors – send messages directly from the browser

Monitoring and Management

- Management Server
- Web based Akka dashboard
- Lots of statistics!
- Alerts and threshold levels for triggering
- Integration with VisualVM, Cacti and Nagios
- Runtime management of local and remote nodes
- Consolidated logging from all nodes
- JMX and SNMP agents on each node
- JMX, SNMPÅ and REST Apis to the management server

Support and Samples

- FAQ secured minor releases with bug fixes and features
- Back-porting of severe bugs to older releases
- Issue tracking of Akka Enterprise
- Advanced Enterprise Samples
- Mail, chat and phone support
- SLA - start working within 4 business hours
- Support CET 8:00-17:00 (24/7 on request)
- Indemnification

On site training

Drottninggatan 108, 113 60 Stockholm ++ +46 733 777 123 ++ jonas@jonasboner.com

allinea

ddt

distributed debugging tool

Allinea DDT – A revolution in debugging

The Distributed Debugging Tool is a comprehensive graphical debugger for scalar, multi-threaded and large-scale parallel and multi-core applications.

DDT puts you in control of your application, whether you are working with a workstation or a thousand processor high performance cluster

DDT has the industry's best parallel debugger interface; one screen to control thousands of processes

Supports all major MPIs, Open MP and queueing systems

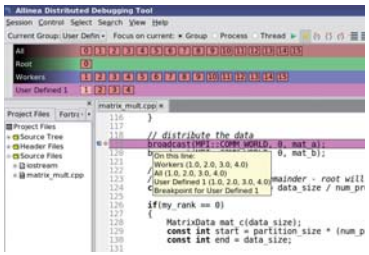
Sophisticated memory debugger finds memory leaks and common errors

Easy-to-use, powerful and cost-effective

Advanced C, C++ and Fortran support

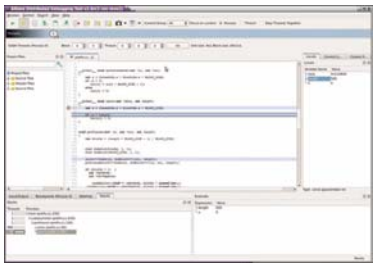
DDT saves time, frustration and money

NVIDIA CUDA SDK 3.0 support



Intuitive Interface

DDT has been designed to make debugging even the most demanding parallel applications easier, faster and more efficient



GPU thread support

DDT allows users to debug applications by both processes and threads

allinea
SCALE TO NEW HEIGHTS

E: info@allinea.com **www.allinea.com**

DOWNLOAD A FREE EVALUATION COPY FROM OUR WEBSITE

distributed debugging tool

Scalable Versatile Intuitive

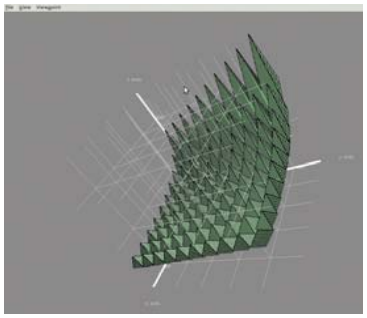
- Run and step groups of processes and threads with visual feedback, including an enhanced parallel stack view for a current overview of process locations
- Compare values across groups of processes and threads: statistically, graphically and automatically by equivalence
- Message queue analysis to detect program deadlock
- Attach to running processes, launch MPI jobs through GUI, or let DDT submit your job to your favourite batch scheduler
- Advanced Fortran and C++ support, including a Fortran module browser and a C++ standard template wizard
- Exceptional memory debugging features and instant detection of out-of-bounds writes and reads for allocated data
- Navigate through local variables, stack frame and complex data structures with ease
- Visualize slices of multi-dimensional arrays using OpenGL graphics
- Automatic display of project source files with syntax highlighting
- OpenMP and multi-threaded support
- Multi Dimensional Array Viewer
- Checkpointing support (gdb, OpenMPI/BLCR)
- Multiple executable debugging support (client/server)

COMPATIBILITY

- Available for AIX, Linux, Solaris and CLE
- Support for the latest processors and platforms: X86, X86-64, NEC SX, IA64, Power, UltraSPARC, IBM Cell BE, CRAY XT and Blue Gene/P
- Compatible with compilers from all major vendors including: Absoft, IBM, Intel, Pathscale, PGI, Sun and GNU

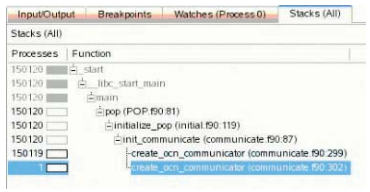
NEW in DDT 2.6

- NVIDIA CUDA 3.0 SDK support
- Qt4 user interface
- Enhanced scalability



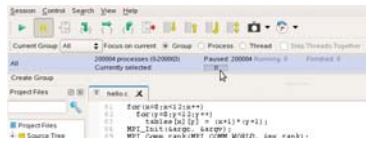
3D OpenGL visualization

DDT allows enhanced array visualization



Enhanced parallel stack view

DDT now shows exactly where your program is, in one single view



Summary Process View

DDT has the ability to control thousands of processes from a single clear panel

allinea

E: info@allinea.com **www.allinea.com**

UK Headquarters: Allinea Software The Innovation Centre Warwick Technology Park Gallows Hill Warwick CV34 6UW
T: +44(0)1926 623 231 F: +44(0)1926 623 232

USA: Allinea Software Inc.
2033 Gateway Suite 500 San Jose Ca. 95110
T: 408 884 0282 F: 408 583 4131

ddt

DDT-2010-06

criticalblue

prism

sequential to parallel migration

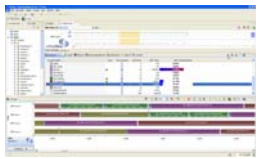
Wondering what benefits multicore platforms would deliver for your existing sequential software? You are not alone. Software teams resist parallelizing their applications because they can't risk potentially complex and uncertain development cycles without real assurance that they can achieve high quality code, on time, with predicted power and performance improvements. By working closely with multicore engineers, CriticalBlue developed Prism™ to take developers from 'what-if' to 'requirements met', streamlining sequential to parallel programming in five best practice steps:

1. Analyze – characterize a serial or parallel application on real workloads. Identify program hotspots, call trees, and data dependencies which will shape achievable parallelism.

2. Explore – before modifying any code, explore different parallel scenarios. What if this function ran in a separate thread? What if these dependencies were removed? Understand the benefits and select the best strategy.

3. Code – Implement the chosen parallelization strategy. Use existing development tools without change.

4. Verify – Confirm that the implementation safely achieved the desired results. Functions properly threaded and dependencies removed? Any potential data races? Synchronization bottlenecks?



5. Tune – Analyze opportunities for further parallelization by reapplying the previous steps.

criticalblue

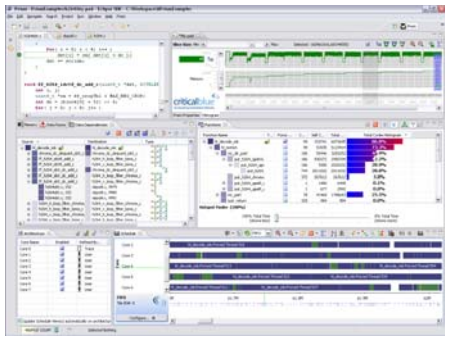
prism

Benefits

Gain more insight, waste less time – analyze multiple parallelization scenarios before making any code changes. Verify parallel objectives are properly achieved.

Augment the way you already work – there are no new languages or proprietary extensions to learn. Start with existing serial or parallel code – mix standard C, C++, assembly, or even binary objects. Work within your own code base. Use familiar development tools and libraries. Prism integrates with Eclipse for maximum portability.

Visualize schedules and pinpoint dependency relationships – parallel programming is about relationships. Understand synchronization between threads, and go directly to the source to see dependency relationships which must be properly maintained.



Design for scalability and reuse – develop parallelization which scales across varying number of cores. Eliminate potential race conditions which could fail as data sources and processing resources change.

Apply precision when needed – platform support ranges from instruction-accurate application analysis to highly specialized and system-accurate chip and board support packages.

Boost your parallel IQ – Prism includes integrated documentation and practical case studies. Supplemental services are available to instill practical parallel practices or augment development teams.

Platform Support Packages

Prism is engineered to work with different processors, programming models, and system environments. Prism capabilities are enabled through platform support packages (PSPs).

Core PSPs are designed to support general multicore architectures and are intended for quick application parallelization. Both ARM and MIPS processors are supported with more on the way.

System PSPs enhance Prism for production processors, chips, and boards. Platform analysis extensions might include configurable full system modeling, power analysis, custom scheduling, specialized parallel programming models, and operating system and board level integration. The Toshiba Venezia PSP is an example which supports Venezia's proprietary threading API and cache coherence validation.

Pricing

Prism pricing starts at $200/month with one core PSP and a minimum one year license. Additional open source or commercial simulators may be required. System PSPs are developed in collaboration with chip and system providers and are priced individually.

Prism is available directly from CriticalBlue or through platform providers.

Evaluation

Wondering what multicore could do for your software? Have a project ready for multicore? Need to learn how to go parallel?

Evaluate Prism free for 30 days at:

http://www.criticalblue.com/prism/eval

System Requirements

- PC with at least 1GB memory
- Windows or Linux operating system

Accelerating Embedded Software

offload

Codeplay's Offload™ is a powerful tool suite combined with a simple programming model to easily and quickly offload large sections of a complex software application onto different processor cores on a multicore CPU. Offload™ lets the compiler do the hard and tedious work offloading the code so that the programmer can concentrate on the application.

Offload requires very little modification to the original source code. It *offloads* a section of code from a normal CPU to an accelerator processor.

```
int ppu_function () {
  int x; // x is now in shared memory
  __offload {
    int y; // y is now in local memory
    y = f (&x, &y);
     /* 'f' is duplicated and called with
       a shared-memory-pointer and
       a local-memory-pointer */
  }
}
```

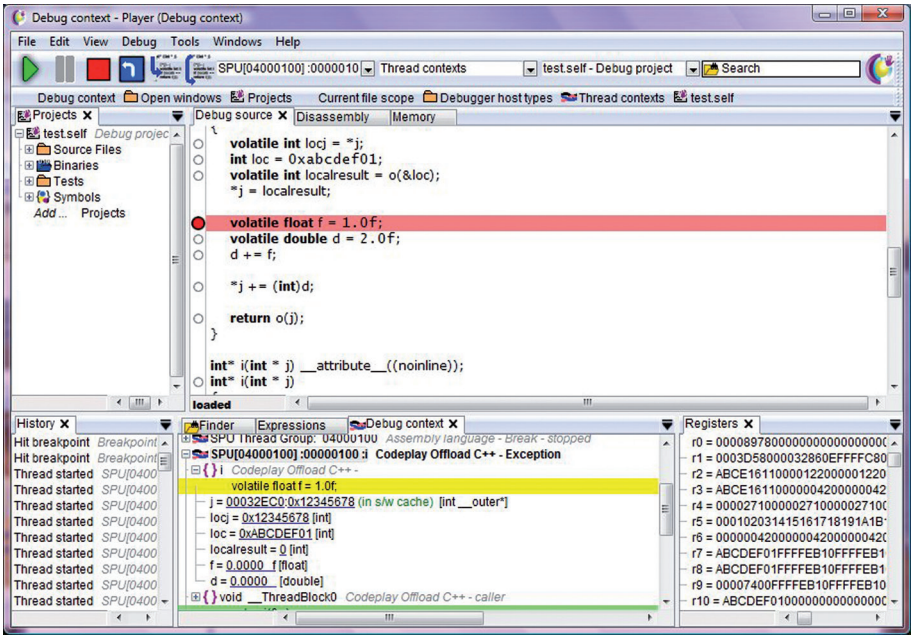
Offload™ is based on Codeplay's Sieve™ System, an award-winning system for taking existing C++ software and transforming it so that it can be compiled with multiple different C compilers and distributed across multiple homogeneous or heterogeneous processor cores.

At the heart of Offload™ lies Codeplay's patented Call-Graph Duplication technology. The Offload™ tool automatically duplicates functions in the Call-Graph that are executed on different types of processors, and intelligently adapts each function so that the data storage and movement is handled correctly for the targeted processor. This removes the need to write the same function differently for the features of each particular processor core, saving the programmer a lot of time and reducing the amount of problems that are likely to arise.

Offload™ enables an **incremental** and **non-disruptive** migration of code to multicore. Just port your application in small, **manageable** steps and **instantly** see the result. This makes Offload™ ideal for porting large legacy codebases. Your application stays written in standard C++ and can be compiled by standard C++ compilers.

With Offload™ your applications can be fully multi-core capable without the disruption, extra development time and inflated development costs you may otherwise encounter.

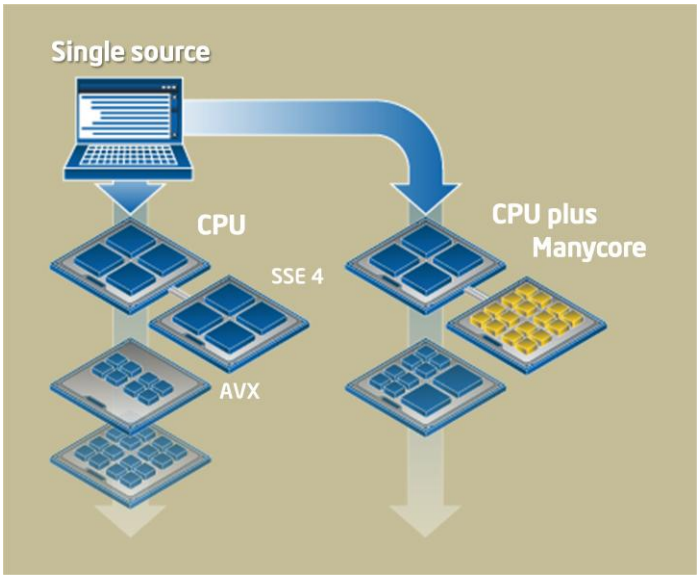
For many applications, simply by offloading a section of code from the host processor to an accelerator processor using Codeplay Offload™, a user can gain a 2.5x performance boost for very little work.



Offload™ Debugger

Intel® Ct Technology

Beta Release Product Brief



From single C++ source, Intel® Ct Technology targets the latest CPUs plus future manycore and hybrid architectures.

Overview

Intel® Ct technology provides a generalized data parallel programming solution that frees application developers from dependencies on particular low-level parallelism mechanisms or hardware architectures.

It produces scalable, portable, and deterministic parallel implementations from a single high-level, maintainable, and application-oriented specification of the desired computation.

It is ideal for applications that require data-intensive mathematical computations such as those found in medical imaging, digital content creation, financial analytics, energy, data mining, science and engineering.

Features

Productivity

- C++ language extensions → compatible with standard compilers
- Thread safe by default → maintainable

Performance

- Scalable and efficient (SIMD + threads)
- One specification → multiple HW-optimized implementations
- Eliminates modularity overhead of C++

Portability

- High-level abstraction (C++ 1st, others later)
- Hardware independent
- Forward scaling

Compatibility

Intel® Ct technology integrates with and complements other Intel developer and parallel programming tools. Presenting a standard C++ library interface, it's also equally compatible with the Microsoft Visual C++ and GCC C++ compilers.

Future Scalability

Applications written today with Intel® Ct technology will automatically scale to support tomorrow's multicore and manycore processors, thereby protecting your development investment.

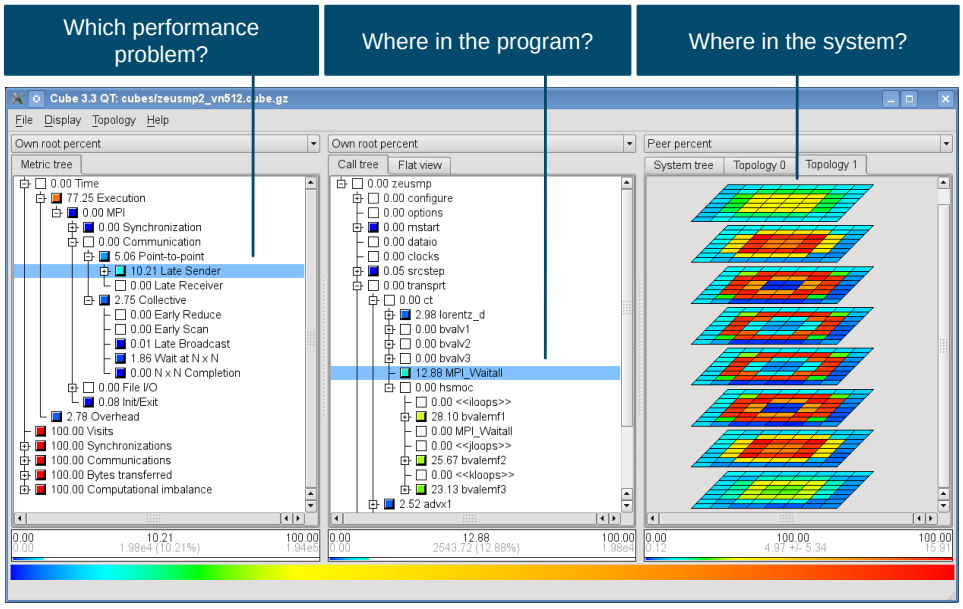
Find out more and sign up for the public beta at http://software.intel.com/en-us/data-parallel/

scalasca

Scalasca is a toolset that can be used to analyze the performance behavior of parallel applications and to identify opportunities for optimization. Target applications include simulation codes from science and engineering written in C, C++ and Fortran and based on the parallel programming interfaces MPI and/or OpenMP. Scalasca has been specifically designed for use on large-scale systems including IBM Blue Gene and Cray XT, but is also well suited for a wide range of small- and medium-scale HPC platforms. The software is available for download under the New BSD open-source license.

Main features:

Scalasca integrates runtime summaries, which are suitable to obtain a performance overview, with in-depth studies of concurrent behavior via event tracing. The traces are analyzed to identify wait states that occur, for example, as a result of unevenly distributed workloads. Especially when trying to scale communication-intensive applications to large processor counts, such wait states can present serious challenges to achieving good performance. The performance behavior can be interactively explored in a graphical browser.



Further information:

- Web site and download: `http://www.scalasca.org`
- Literature: M. Geimer, F. Wolf, B. J. N. Wylie, E. Ábrahám, D. Becker, B. Mohr: The Scalasca performance toolset architecture. Concurrency and Computation: Practice and Experience, 22(6):702-719, April 2010.

Scalasca is a joint project of:

JÜLICH FORSCHUNGSZENTRUM

German Research School for Simulation Sciences

PVS-Studio

PVS-Studio is a toolset of static source code analyzers designed for detecting 64-bit and parallel errors in software. The tool is designed for developers of state-of-the-art resource-intensive applications. PVS-Studio is integrated in Visual Studio 2005/2008/2010 environment and supports functioning in C/C++ languages. PVS-Studio comprises Viva64 analyzer for detecting errors in 64-bit programs and VivaMP analyzer for detecting errors in parallel programs built with the help of OpenMP technology. Estimation of time and complexity of code migration process to 64-bit systems is an important feature of the tool.

PVS-Studio tool is the own project of Russian company OOO "Program Verification Systems".

PVS-Studio Features

- Visual Studio 2005/2008/2010 integration;
- online help;
- pdf documentation;
- load/save analysis reports;
- command line version available;
- work on all cores/processors;
- 32-bit projects preliminary verification:
- cost estimation of code migration to 64-bit;
- support of Windows (LLP64) and Linux (LP64) data models;
- interactive filters;
- easy integration into team development process;
- marking sources to check new code only.

Bug Types Detectable by PVS-Studio

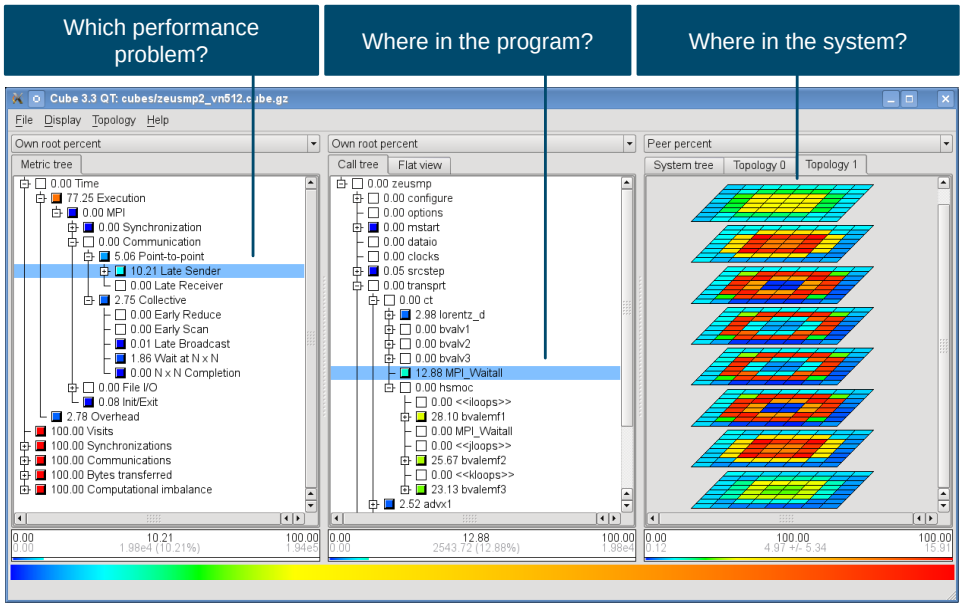
- errors occurring during migration of 32-bit applications to 64-bit systems;
- errors occurring during processing of more than two gigabytes of data;
- 64-bit errors in new code developed without consideration of Win64 architecture special features and LLP64 data model special features;
- ineffective use of memory in 64-bit applications;
- race condition errors occurring in parallel OpenMP-programs;
- errors due to lack of understanding of parallel algorithms building on the basis of OpenMP technology;
- use of ineffective OpenMP constructions.

scalasca

Scalasca is a toolset that can be used to analyze the performance behavior of parallel applications and to identify opportunities for optimization. Target applications include simulation codes from science and engineering written in C, C++ and Fortran and based on the parallel programming interfaces MPI and/or OpenMP. Scalasca has been specifically designed for use on large-scale systems including IBM Blue Gene and Cray XT, but is also well suited for a wide range of small- and medium-scale HPC platforms. The software is available for download under the New BSD open-source license.

Main features:

Scalasca integrates runtime summaries, which are suitable to obtain a performance overview, with in-depth studies of concurrent behavior via event tracing. The traces are analyzed to identify wait states that occur, for example, as a result of unevenly distributed workloads. Especially when trying to scale communication-intensive applications to large processor counts, such wait states can present serious challenges to achieving good performance. The performance behavior can be interactively explored in a graphical browser.



Further information:

- Web site and download: `http://www.scalasca.org`
- Literature: M. Geimer, F. Wolf, B. J. N. Wylie, E. Ábrahám, D. Becker, B. Mohr: The Scalasca performance toolset architecture. Concurrency and Computation: Practice and Experience, 22(6):702-719, April 2010.

Scalasca is a joint project of:

JÜLICH FORSCHUNGSZENTRUM

German Research School for Simulation Sciences

ENTERPRISE ARCHITECT

SPARX SYSTEMS

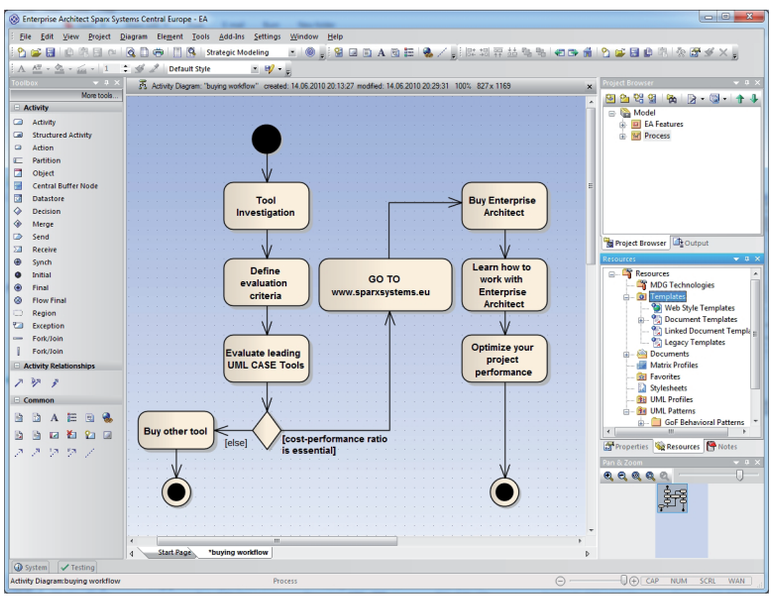
Unleash your potential with Enterprise Architect!

Enterprise Architect is the UML-CASE-Tool of Choice for more than a quarter-million of System Analysts, Software Designers and Architects, Programmers, Quality Managers, Project Managers and Support Representatives.

EA provides full life cycle modeling for:

- Business and IT systems
- Software and Systems Engineering
- Real-time and embedded development

With built-in requirements management capabilities, Enterprise Architect helps you trace high-level specifications to analysis, design, implementation, test and maintenance models using UML,SysML,BPMN and other open standards. So don't hesitate and check our offerings!



Due to the focus of SparxSystems on Enterprise Architect and the vast experience with UML Modeling, we are able to continuously provide our customers with a state-of-the-art tool at a reasonably good price! In addition we offer a free update for 12 months.

For actual License Costs see www.sparxsystems.eu for further information.

SPARX SYSTEMS Central Europe

SparxSystems Software GmbH - Central Europe | Ferrogasse 80/3 | 1180 Vienna | Austria
T +43 (0)662 90 600 2041 | **F** +43 (0)662 90 333 3041 | **E** sales@sparxsystems.eu | **W** www.sparxsystems.eu

Microsoft Visual Studio 2010

Visual Studio 2010 and the .NET Framework 4 are major updates to Microsoft's development platform with additions and enhancements across almost all areas, such as the integrated development environment (IDE) and new testing and modeling tools. New tools and technologies address emerging computing trends such as parallelism.
Visual Studio 2010 provides powerful tools to support the development of software for modern multicore and multiprocessors hardware. Further it covers the whole application lifecycle by providing tools for managing projects, maintaining source code and finding and fixing bugs. Testers and developers can use manual and automated testing coupled with advanced debugging tools to ensure they are building the right application, the right way.
Visual Studio supports different programming languages by means of language services, which allow the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C/C++ Visual Basic .NET, C# (via and F#. Support for other languages such as M, Python, and Ruby among others is available via language services installed separately.

Features

- **IntelliTrace™ eliminates 'No Repro':** Easily step through code that was previously executed on the same or another machine in order to identify what happened during the code execution and significantly cut down on time spent reproducing a bug.
- **Understand existing architectures:** The Architecture Explorer and UML sequence diagrams help you explore and understand your existing code assets and their inter-dependencies.
- **Ensure architectural compliance:** Use layer diagraming to define and communicate logical application architecture and optionally enforce architecture rules with check-in policies as code is created.
- **Profiling application performance for multicore and multiprocessor hardware:** Measure the performance of your applications and locate performance bottlenecks in CPU utilization, memory consumption, database interactions, Jscript/Ajax call patterns and concurrency synchronization. Use the performance profiler in conjunction with Load Testing to identify performance bottlenecks under stress and load.
- **Discovering common coding errors:** Code Analysis is now simple to configure with customizable rule sets that can be targeted at specific scenarios or areas of emphasis. Enforce rule sets with the code analysis check in policy to reduce common coding errors in application or database code before they get into production.
- **Prototyping user interface ideas quickly:** By using SketchFlow in Expression Studio you can quickly deliver a functioning prototype that looks and feels like handwritten mockups.
- **Automating user interface testing:** Use Coded User Interface Tests to automate the testing of UI elements in your applications. Visual Studio 2010 automatically generates test code that can be executed manually or incorporated as part of your build process to automate UI regression testing.
- **Creating, managing and executing tests:** Microsoft Test Manager 2010 lets you easily define your testing effort for a specific iteration in your project, run manual tests and measure your progress. In addition to creating and managing test assets like plans, suites and test cases you can also create and manage virtual lab configurations for your test environments.

Additional information:

http://www.microsoft.com/visualstudio/en-us/

The widespread availability of multicore processors represents a significant challenge for software developers. Software has to be parallelized in order to take full advantage of the potential of having multiple cores ready for parallel processing.

The goal of the MWare project is the research and development of methods, techniques and tools for the efficient design and implementation of parallel software for the multicore processors of the future. The research is based on important applications that are already available from the Fraunhofer institutes involved in the project.

This market overview provides a survey of tools for multicore software development.

Developers use these tools to find parallelism in their applications, to implement parallel solutions and to tune for optimal performance.

For further information please visit
http://www.mware.fraunhofer.de/EN/

ISBN 978-3-8396-0165-5