4-page Case Study. Published on:
http://www.microsoft.com/casestudies/Case_Study_Detail.aspx?casestudyid=4000006833

Rate This Evidence:


Composite C1 Content Management Solution Uses Parallelization to Deliver Huge Performance Gains

Content management system vendor Composite needed to parallelize its software to realize the performance gains enabled by today's multicore processors. Composite took advantage of the new parallel-programming tools provided in the Microsoft Visual Studio 2010 development system and the .NET Framework 4 to parallelize its code. The company's efforts have yielded impressive performance gains: An eight-core server is delivering a 60 percent reduction in page-rendering times and an 80 percent reduction in the compilation of dynamic types upon system initialization. What's more, by using the latest Microsoft aids for parallel programming, Composite was able to implement parallelism in its solution quickly and cost-effectively, with very little developer effort.

Situation

Microsoft Gold Certified Partner Composite develops and sells Composite C1, a content management system (CMS) designed to help companies build Web sites that combine solid marketing infrastructure, innovative design, and strong usability. Originally founded as a Web development shop, Composite decided to build its second-generation CMS in 2005 after realizing that the Microsoft

> "With Visual Studio 2010 and the .NET Framework 4, Microsoft is providing tools that immensely simplify parallel development. Developers can simply 'declare intent' to do parallelization and leave it to the underlying framework to"

handle the rest.
Marcus Wendt
Cofounder and
Product Manager,
Composite

Visual Studio 2005 development system and the Microsoft .NET Framework 2.0 presented an opportunity to build a solution that could meet the needs of both Web developers and designers.

As new versions of Visual Studio and the .NET Framework become available, Composite examines them closely to determine how they can be applied to improve its own product. For example, the company took advantage of Visual Studio 2008 and the .NET Framework 3.5 to add support for Language-Integrated Query (LINQ) in version 1.2 of Composite C1. Composite began this exercise again in 2009, when it joined an early adopter pro-gram for Visual Studio 2010 and the .NET Framework 4 and began planning for the development of Composite C1 version 1.3.

One area on which Composite decided to focus was performance—specifically, how to optimize its software to get the most out of modern, multicore processors and multiprocessor servers. “For the past few decades, we’ve all benefited from rapidly increasing processor clock speeds,” says Marcus Wendt, Cofounder and Product Manager at Composite. “However, this extended ‘free lunch’ is over, in that clock speeds have leveled off and chip manu-facturers are turning to multiple-processor cores for further gains in processing power. Therein lies the challenge, in that most applications today—Composite C1 version 1.2 included—are not multicore optimized, which can result in one core running at 100 percent while the rest remain idle. To get the best performance out of today’s multicore processors, we needed to introduce parallelization into our code.”

Solution

Composite took advantage of the new parallel-programming tools provided in the Microsoft Visual Studio 2010 development system and the .NET Framework 4, and the company’s efforts have yielded significant performance gains in multiple areas. “Parallel programming has traditionally been difficult, tedious, and hard to debug, with very limited tool support,” says Wendt. “The System.Threading namespace in the .NET Framework has existed for years, but in the past it required a lot of ‘plumbing code’ to use effectively. Visual Studio 2010 and the .NET Framework 4 eliminate a lot of complexity to make parallel programming much easier.”

New Parallel-Programming Aids

Composite parallelized its code by using Visual Studio 2010 and the new parallelization libraries in the .NET Framework 4, including:

- **Task Parallel Library (TPL),** which includes parallel implementations of for and foreach loops (For and For Each in the Visual Basic language), as well as rich support for coordinating the asyn-chronous execution of individual tasks. Implemented as a set of public types and APIs in the System.Threading.Tasks namespace, TPL relies on an extensible task-scheduling system that is integrated with the .NET ThreadPool and scales the degree of concurrency dynamically, so that all available processors and process¬ing cores are used most efficiently.
- **Parallel Language-Integrated Query (PLINQ),** a parallel implementation of LINQ to Objects that combines the simplicity and readability of LINQ syntax with the

power of parallel programming. PLINQ implements the full set of LINQ standard query operators as extension methods in the System.Linq namespace, along with additional operators to control the execution of parallel opera-tions. As with code that targets the Task Parallel Library on top of which PLINQ is built, PLINQ queries scale in the degree of concurrency according to the capabilities of the host computer.

- **New Data Structures for Parallel Programming,** which include concurrent collection classes that are scalable and thread safe; lightweight synchronization primitives; and types for lazy initialization and producer/consumer scenarios. Developers can use these new types with any multithreaded application code, including that which uses the Task Parallel Library and PLINQ.

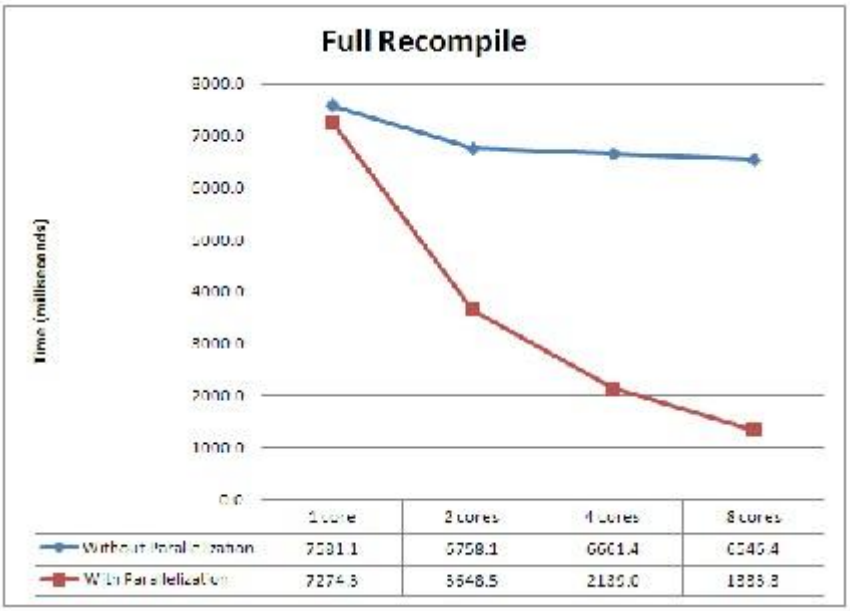

Figure 1. Parallelism improves performance in the compilation of dynamic types

Composite also took advantage of the new Parallel Stacks and Parallel Tasks windows for debugging code, which are provided in Visual Studio 2010 Ultimate, Premium, and Professional. Visual Studio 2010 Premium and Ultimate also have a new Concurrency Visualizer, which is integrated with the profiler to provide graphical, tabular, and numerical data about how multithreaded applications interact with themselves and with other programs. “The Concurrency Visualizer and other parallel-programming tools in Visual Studio 2010 are a great help in that they enable developers to quickly identify areas of concern and navigate through call stacks and to relevant call sites in the source code,” says Martin Ingvar Jensen, Senior Developer at Composite.

Faster Compilation of Dynamic Types

In parallelizing its code, Composite identified two areas that were prime candidates for concurrency. One area was the compilation of dynamic types, which is done upon reinitializing Composite C1 and can take up to 90 seconds for large Web sites. “Our dynamic type system enables developers to

“Parallel programming has ”

> traditionally been difficult, tedious, and hard to debug, with very limited tool support.… Visual Studio 2010 and the .NET Framework 4 eliminate a lot of complexity to make parallel programming much easier.
> Marcus Wendt
> Cofounder and Product Manager, Composite

design data types using our Web interface and treat them as 'real' .NET Framework types, such as executing LINQ statements against them," says Wendt. "This is achieved by generating and compiling code upon initialization, which is time-consuming and processor intensive. Without parallelization, the CPU Usage History indicator would show one core running at 100 percent utilization while the other cores were doing no work at all."

Parallelization was achieved by changing a classic foreach loop in the application's compilation manager to a Parallel.ForEach loop—a task that required changing three lines of code. The performance gains achieved through this effort are shown in Figure 1. "The time that Composite C1 spends compiling dynamic types has been significantly reduced on multicore systems, with performance increasing steadily as more cores are available," says Wendt. "Not only does this reduce startup times upon initial deployment, but it also makes developers more efficient and productive when they're working with our software.

"Given the very limited amount of code work we had to do, it's fair to say that the support for parallel programming provided in the .NET Framework 4 worked very well for us," says Wendt.

Improved Page-Rendering Performance

Composite also saw the potential for big performance gains in page rendering. "In Composite C1, the rendering process handles the construction of page elements such as navigation aids, news listings, search results, and so on," explains Wendt. "In the past, these dynamic page elements were rendered serially, one after the other. Web pages often contain multiple renderings, so the ability to perform those operations in parallel provides a huge opportunity to improve performance—and thus deliver a better end-user experience."

The company parallelized the rendering process by using the new data structures for parallel programming. A code sample is shown in Figure 2; Figure 3 shows the performance gains. "Rather than using a C# statement such as foreach to declare that concurrency is desired, we call the static ForEach method on the parallel class, passing a collection of data and a lambda expression you want to execute," explains Wendt. "The .NET Framework 4 handles all of the complex thread management in accordance with the underlying hardware platform, firing off more threads as more cores are available. This is work that most developers will happily let the underlying programming framework handle—in a way that's likely more efficient and optimized than what they could implement by hand."

```
Without Parallelization (.NET Framework 3.5)

List<XhtmlDocument> presentations = new List<XhtmlDocument>();
foreach (var p in DataFacade.GetData<Product>())
{
    presentations.Add(MakePresentation(p));
}


With Parallelization (.NET Framework 4)

ConcurrentQueue<XhtmlDocument> presentations =
    new ConcurrentQueue<XhtmlDocument>();
Parallel.ForEach(DataFacade.GetData<Product>(), p =>
{
    presentations.Enqueue(MakePresentation(p));
});
```

Figure 2. Use of the new parallelization libraries in the .NET Framework 4 to improve page-rendering performance

Composite's use of ConcurrentQueue<T> instead of List<T> to store calculation results is also noteworthy. "We do this because List<T> is not thread safe, meaning that you need to add locking to your code or brace yourself for some unexpected results at runtime," explains Wendt. "Developers still need to think about thread safety, but the .NET Framework 4 makes the coding much easier, transforming the process from 'be very careful and do the hard work' to just 'be careful.' "

Other Useful New Features and Capabilities

Beyond parallelization, Composite devel-opers are finding Visual Studio 2010 and the .NET Framework 4 useful in other new ways, such as application lifecycle manage-ment. "We're planning to upgrade our current Visual Studio Team System 2008 Team Foundation Server implementation to Visual Studio Team Foundation Server 2010, so we can take advantage of the new Scrum templates," says Wendt. "There are a lot of great new features in the new Scrum templates provided with Visual Studio Team Foundation Server 2010, and we're expecting that they will be of great benefit to the development process."

"Developers still need to think about thread safety, but the .NET Framework 4 makes the coding much easier, transforming the process from 'be very careful and do the hard work' to just 'be careful.' "

Marcus Wendt
Cofounder and

Product Manager, Composite

Developers also are taking advantage of the new support for covariance in generics. “Our system uses interfaces as the generic parameter when querying data from our data layer,” explains Wendt. “Because Visual C# 3.5 did not support covariance, we had to do a lot of expression tree transforma-tion when using LINQ to SQL. Generic covariance should give us a performance boost, make our code simpler and thus easier to maintain, and enable us to add ‘data schema inheritance’ to our data layer to enable some pretty interesting new features.”

Benefits

By taking advantage of the new parallel-programming aids provided in Visual Studio 2010 and the .NET Framework 4, Composite was able to easily capitalize on the performance gains enabled by modern multicore processors. “With Visual Studio 2010 and the .NET Framework 4, Microsoft is providing tools that immensely simplify parallel development,” says Wendt. “Devel-opers can simply ‘declare intent’ to do parallelization and leave it to the under-lying framework to handle the rest. The process isn’t foolproof in that developers still need to understand parallel program-ming, but it’s quite easy to use and, when used correctly, can enable applications to utilize modern microprocessors much more efficiently.”

Up to 80 Percent Reduction in Processing Time

Composite’s use of parallel programming is delivering significant performance gains, in turn improving the Composite C1 user experience for both Web developers and Web site end users. “On a server configured with two quad-core processors, our use of parallel programming delivered an 80 per-cent reduction in the time required to com-pile dynamic types, which means that Web developers don’t need to wait as long when working with Composite C1,” says Wendt.

The test that Composite constructed to measure the effects of parallelization on page-rendering times showed a decrease of more than 60 percent. “Our use of parallelization reduced the time required to render a test Web page containing eight functions from 115 to 40.5 milliseconds—even with one page element that takes 40 milliseconds to render on its own,” says Wendt. “That’s the beauty of parallelization, in that it enables us to break up a chunk of work into independent tasks and execute them concurrently to get the job done faster.”

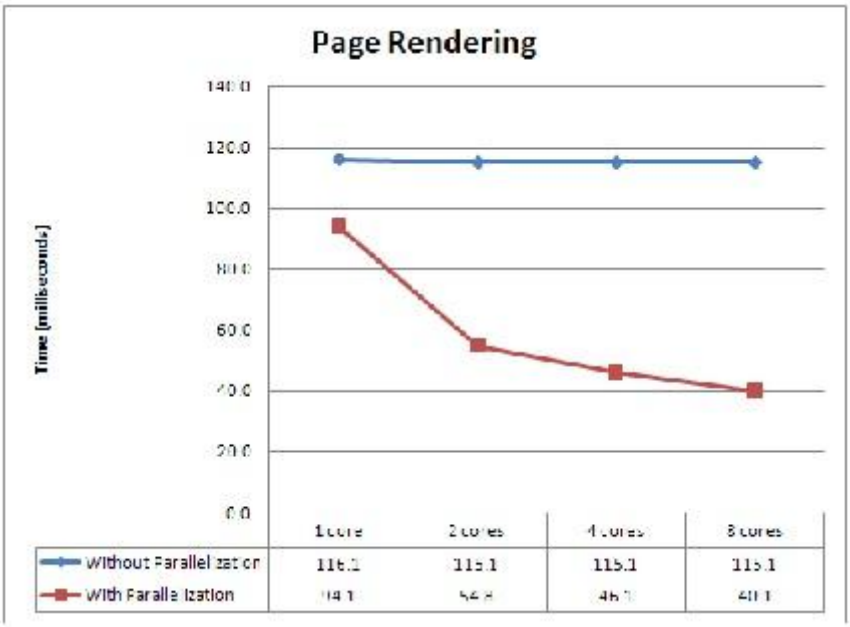

Figure 3. Page-rendering performance with and without parallelization

Minimal Developer Effort Required

Composite was able to implement parallel¬ism in its application quickly and cost-effectively, with very little developer effort. “The hardest part was figuring out where we could benefit from parallelization—an area where the Concurrency Visualizer in Visual Studio 2010 was very helpful,” says Wendt. “Visual Studio 2010 and the .NET Framework 4 extend the ‘free lunch’ enabled by increas-ing processor speeds over the past few decades with an ‘almost free lunch’—achieved through a set of tools that make it far easier and more practical to imple-ment parallelization. The new parallel-programming aids provided by Microsoft aren’t a ‘magic wand’ that will make existing code run in parallel by itself, but they do make the work required a whole lot easier.”

Microsoft Visual Studio 2010

Microsoft Visual Studio 2010 is an integrated development system that helps simplify the entire development process from design to deployment. Unleash your creativity with powerful prototyping, modeling, and design tools that help you bring your vision to life. Work within a personalized environment that helps accelerate the coding process and supports the use of your existing skills, and target a growing number of platforms, including Microsoft SharePoint Server 2010 and cloud services. Also, work more efficiently thanks to integrated testing and debugging tools that you can use to find and fix bugs quickly and easily to help ensure high-quality solutions.

For more information about Visual Studio 2010, go to www.msdn.microsoft.com/vstudio

Microsoft Parallel Computing Platform

Developers today face an unprecedented opportunity to deliver new software expe-riences that take advantage of multicore and many-core systems. Microsoft is taking a comprehensive approach to simplifying parallel programming, working at all levels of the solution stack to make it simple for both native-code and managed-code developers to safely and productively build robust, scalable, and responsive parallel applications.

For more information on the Microsoft Parallel Computing Platform, go to:
http://msdn.microsoft.com/en-us/concurrency/default.aspx

For More Information

For more information about Microsoft products and services, call the Microsoft Sales Information Center at (800) 426-9400. In Canada, call the Microsoft Canada Information Centre at (877) 568-2495. Customers in the United States and Canada who are deaf or hard-of-hearing can reach Microsoft text telephone (TTY/TDD) services at (800) 892-5234. Outside the 50 United States and Canada, please contact your local Microsoft subsidiary. To access information using the World Wide Web, go to:
www.microsoft.com

For more information about Composite, visit the Web site at:
www.composite.net