

"Building the most robust threading model possible proved to be a real challenge. We welcomed the help we got from the Intel® Software Partner Program, which also helped us quantify our success afterward."

- Bill Wise, Chief Executive Officer,
Pocket Soft, Inc.

Increasing the Pace of Change

Developing for multicore helps Pocket Soft generate data patches with scalable swiftness.



Pocket Soft

CHALLENGE

Produce software patches that contain just the changes between two bodies of data, to make the distribution of updates as efficient as possible. Generate those files quickly, to enable rapid, frequent updating.

SOLUTION

Pocket Soft enabled its RTPatch* Server product to take advantage of multicore processors. That advance overcomes many of the former limitations posed by the computationally-intensive, typically processor-bound nature of the application workloads.

CUSTOMER BENEFIT

Dramatically Improved performance helps control increasing time requirements to handle growing bodies of data and allows for a broader range of usage models. Because the multicore version of Pocket Soft RTPatch Server is a drop-in replacement for previous versions of the product, no integration effort is required on the part of Pocket Soft's existing customers.

More

Learn more: www.intel.com/partner

PROOF POINT

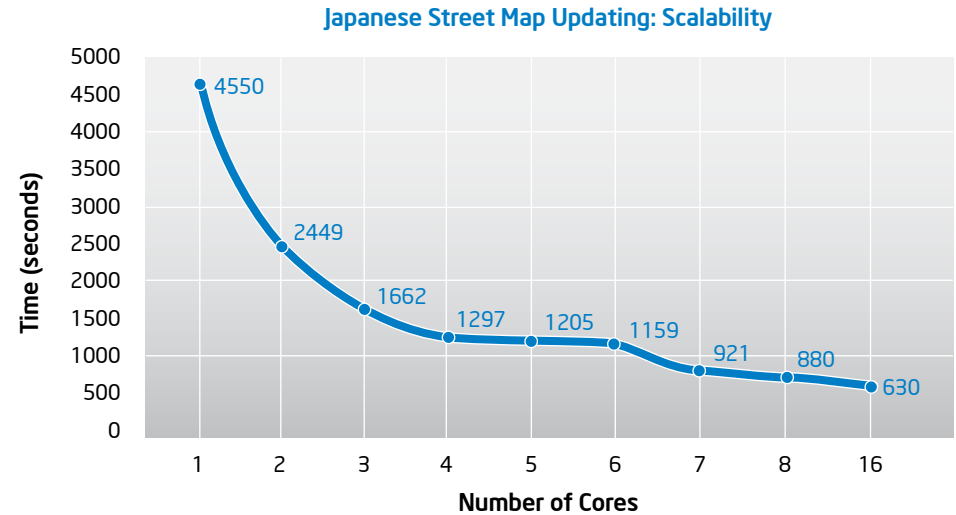
Updating Japanese Street Maps

A Japanese company that distributes electronic address data was looking for a solution to make it more efficient to distribute updates. The byte-level differencing technology offered by Pocket Soft RTPatch* Server proved to be well suited to the challenge.

Creating a file that includes just the changes between two versions of the data set allows the distributed information to be more compact. Unfortunately, generating these patches is very processor-intensive, and early results took nearly 70 hours, which was unacceptable for frequent updates.¹

Pocket Soft refined its algorithms, and each version of Pocket Soft RTPatch Server is more efficient than the last. With Pocket Soft RTPatch Server 5.0, its most recent release, the industry has its first multicore-enabled product in this category.

Scalability across multicore processors has reduced the time requirement, and currently it takes about 10 minutes to generate a patch consisting of all the updates to the Japanese address data,¹ as shown in the chart, "Japanese Street Map Updating: Scalability."



Pocket Soft RTPatch* Server scales byte-level differencing among multiple processor cores.¹

The Value of Discovering Differences

Delivering updates to a piece of software or other set of data is more efficient if the updates are limited to just what has changed, instead of replacing the entire data set. Of course, that efficiency requires the ability to determine what the differences are, and to be viable, the process must itself be efficient.

The more granular the comparisons are between two data sets, the more completely Pocket Soft RTPatch* Server can discover the differences between them, and the smaller the eventual software patch will be. Therefore, creating smaller patches requires more time or computational power, which makes clear the value of computational efficiency to the process.

The task of identifying the very granular variation between two data sets is conventionally called "byte-level differencing." The computational demands of this process on large bodies of data can be immense, and typically, it is almost entirely processor-bound. With the introduction of increasingly high core counts in computer hardware, use of a multi-threading process has clear value, and Pocket Soft has risen to the opportunity.

Meeting a Complex Threading Challenge

Introducing software threading into the byte-level differencing process is more complicated than at first it may appear. It is not simply a matter of performing a similar task repeatedly on many small chunks of data because the discrete comparison tasks are not independent.

Individual threads must cooperate on overlapping pieces of data to find repeated patterns. RTPatch Server also continually performs hashing and indexing operations, and to address their time-consuming nature, the software contains algorithms to make sure specific operations are not repeated. That verification requires additional cooperation among threads.

The Pocket Soft threading project for RTPatch Server was tremendously successful, resulting in the following outcomes:

- **High utilization on all processor cores.** Given sufficiently sized inputs, the application is able to utilize all cores at an average of approximately 90 percent (reaching a steady state of 99 percent is not uncommon).¹

- **Equivalent patch size to that of a single-core version.** The multicore version of RTPatch Server creates patches that are within about one percent in size of those created by the single-core version, without varying as more cores are applied to the task.¹

- **Overall application speed-up.** RTPatch Server 5 scales smoothly on multicore systems, providing build times that are in the range of (n-1)x faster than the single-threaded case, where n is the number of cores used.¹

- **Able to benefit from Intel® Hyper-Threading Technology when available.** RTPatch Server generates an extra thread so that Intel Hyper-Threading Technology enables the application to utilize spare cycles on otherwise idle cores.¹

Spinning Off the Threading Technology

Recognizing the value of creating the most robust threading model possible, the team at Pocket Soft undertook a detailed examination of the challenges and opportunities associated with enabling RTPatch Server for multicore processors. As part of that effort the team distinguished between three different options, characterized as follows:

- **Low-level multi-threading.** This model, which consists of mostly compiler-assisted threading of loops and other iteration constructs, is relatively simple but tends not to fully utilize processor resources, limiting speed-up potential.
- **High-level multi-threading.** Dividing the main task up into distinct phases and pipelining or overlapping them is also relatively simple but not applicable to many classes of problems.
- **Mid-level multi-threading.** Threading subtasks that are more substantial than a local loop, but less substantial than an entire program phase, is a more complex effort but can potentially deliver very high resource utilization and be broadly applied.

On the basis of these observations, Pocket Soft recognized that the potentially high payoff in adopting the mid-level multi-threading model justified its added complexity and effort. The development team chose to approach the challenge in a generalized way, abstracting it away from the specific compute problems associated with byte-level differencing.

The team successfully created a tool set to support robust multi-threading based on an API with supporting DLLs and a static library. Having met its goal of creating a solution that is not limited to the specific product implementation, the company identified the value of marketing those tools as a distinct product offering. As a result, the Postulate5 division of Pocket Soft now offers the UnThread* tool set; more information is available at postulate5.com. This added profit center is a substantial extra benefit to the development effort.

Broad Applicability across Industries

Perhaps the most intuitive and familiar example of byte-level differencing in practice is with software patches. Most users are familiar with the process of downloading updates to their operating systems, antivirus programs, and other applications. To create these updates, the vendor builds a new version of the software or data set and then uses a product such as RTPatch Server to create a downloadable digest that consists of only the changes.

Beyond that common usage model, the robust, multicore-enabled byte-level differencing provided by Pocket Soft RTPatch Server 5 is valuable to a large number of usage models that span many industries. As described on the Pocket Soft Web site, "RTPatch Server may be used wherever information is changed at one location and needs to be updated at another."

Examples of real-world implementations include the following:

- **Distribute airline maintenance manuals.** Technical documentation for a large aircraft fleet changes daily and must be distributed throughout a large system of servers for use by maintenance crews. By updating only the changes to the repository, the company can keep the documentation up to date using relatively low levels of network capacity.
- **Satisfy requests for updated investment information.** A central data repository supports 300,000 clients and more than 200 remote data stores. When a request for updated information is received, RTPatch Server dynamically generates the necessary data set to update the requesting data store, optimizing WAN capabilities and data availability.
- **Remove the need for overnight, international distribution of DVDs.** Branch offices need to provide daily updates of data images in the form of 4 GB ISO files to corporate headquarters. Rather than shipping DVDs, RTPatch Server generates a file of just the differences that is small enough to be sent by e-mail, achieving dramatic savings in cost and effort.

As data volumes associated with the full range of business operations continue to grow, increasingly sophisticated means are needed to handle the distribution

"Historically, there is a trade-off in generating patch files, where spending more time on the calculations lets users generate smaller files. By enabling for multicore Intel® architecture, we can now let our customers create smaller patches within their time constraints."

- Kerry N. Jones,
Chief Technology Officer, Pocket Soft, Inc.

of that data. Information is now a key business asset for many companies, and maintaining the freshness of that information throughout an organization is vitally important. Simply replicating a full data set between sites can represent a large inefficiency, in contrast to the clear advantage offered by byte-level differencing.

Many unrealized possibilities exist for increased efficiency by dramatically reducing the size of data transmissions used by all sorts of businesses, research institutions, and government customers. With the increased ability of RTPatch Server to take advantage of multicore processing power, those opportunities will potentially increase.

A Partnership that Breeds Success

It comes as no surprise that as Pocket Soft was optimizing its software for multicore and building its threading engine, it found particular value from its relationship with the Intel® Software Partner Program. Looking forward, the team anticipates deepening that relationship to further enhance RTPatch Server, aligning it with future generations of hardware platforms:

- **Getting engineering expertise.** Pocket Soft engineers worked directly with their

Intel colleagues and took advantage of the Intel® Software Network Parallel Programming Community to research hardware and receive answers to development questions.

- **Confirming optimization results.** Testing with the Intel® Concurrency Checker rated Pocket Soft's success at optimizing for multicore at the 100th percentile compared to results from its with peers, confirming that the team's work had fully paid off.

- **Taking advantage of co-marketing opportunities.** Pocket Soft uses Intel logos in its marketing, helping emphasize its relationship with Intel and its role as a technology leader.

Looking to the future, Pocket Soft has purchased Intel® Parallel Studio, which it intends to incorporate in future development efforts. Continuing to drive the quality and performance of Pocket

Soft RTPatch Server is a clear imperative for the company that will pay dividends to its customers. Pocket Soft engineers are already developing the solutions to their next round of challenges, so the industry can rest assured that patch technology will continue to push the envelope.

About the Intel® Software Partner Program

The Intel® Software Partner Program provides a framework for collaborative solution development around Intel® architecture. From business planning and product development to marketing and sales, the program helps to drive increased business success and market opportunities.

Learn more at www.intel.com/partner.

Success Story by:



Learn more about
Pocket Soft, Inc.:
www.PocketSoft.com

Visit the Intel® Software
Network Parallel
Programming Community
[software.intel.com/
en-us/parallel](http://software.intel.com/en-us/parallel)

¹ Results reported by Pocket Soft, Inc.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2010 Intel Corporation. All rights reserved. 0510/BM/MESH/PDF