WHITE PAPER

Zircon Adaptive Software on SGI® Altix® UV 1000 for High Performance Data Analytics

Alexander Mintz†, Dr. Douglas C. Schmidt†, Dr. Grigory Vilkov†,
Dr. Jaiganesh Balasubramanian†, Pooja Varshneya†, Dr. Sanhita Sarkar*
Zircon Computing LLC† and SGI*

Abstract

Software developers and architects creating and upgrading their code to run on new processor architectures for embedded market segments are trying to harness the processing power available from these multi-core platforms. These market segments include fraud detection, market risk analysis, digital image processing, storage, and many others. Unfortunately, legacy applications in these domains cannot leverage the parallelism inherent in multi-core systems without extensive remodeling and reprogramming. Despite the price/performance ad-vantages of these systems, it is hard to accelerate and scale the performance of applications to take advantage of the additional processing power.

This paper describes how Zircon software dramatically improves application performance on a SGI® Altix® UV 1000 platform comprising of 512-cores, 4TB memory. Evaluation of Zircon software's capabilities has been done via a case study that parallelized two types of complex financial applications. In addition to analyzing empirical results, this paper shows how Zircon software can leverage the multi-core parallelism and large memory of SGI Altix UV 1000 to derive efficient and scalable solutions to common problems encountered by developers while dealing with parallelization of applications within a single system image.

** 30 Galesi Drive, Suite 202B, Wayne, NJ 07470, USA*
† 46555 Landing Parkway, Fremont, CA 94538, USA

1. Introduction

Parallel application programming is becoming increasingly important to leverage multi-core platform, large memory architectures for complex data analytical applications in a fast and scalable manner. Financial trading applications are a few of the many analytical applications that cannot easily leverage the additional processing power available in multi-core architectures, without substantial modifications to the application code. What is needed, therefore, is a solution that can leverage hardware and software innovations in distributed and parallel computing, while simultaneously minimizing software modifications and reducing the effort needed to incorporate these innovations to accelerate and scale up performance.

To meet this need, Zircon Computing created Zircon software [2], which provides a unique platform to develop and deploy software programs for both colocated and/or distributed parallelism via pattern-oriented APIs and tools. Zircon software's colocated parallelism capabilities can leverage modern multi-core platforms with minimal effort and run-time overhead. Likewise, Zircon software's distributed parallelism capabilities support ultra short latency (e.g., 40 microsecond network roundtrip), automatic service discovery, fault tolerance, service request recovery, and highly efficient point-to-point communication utilizing load balancer with real-time feedback. This paper analyzes the results of experiments conducted in a case study where Zircon software was used to parallelize the following two types of complex financial applications on SGI Altix UV 1000 comprising of 512 cores, 1024 threads and 4 Terabytes (TB) of memory:

- **Applications with data-independent tasks,** such as *Heston stochastic volatility model calibration* [6]. A data-independent task contains several independent subtasks that do not depend on each other for input data, so it can be parallelized easily by running several subtasks concurrently on multi-core processor architectures even in cloud computing environments.

- **Applications with data-dependent tasks,** such as *scenario analysis for path-dependent trading strategies.* Data-dependent tasks contain several data-dependent subtasks, where output of a preceding subtask serves as an input for the next sub-task computation. Data dependent tasks can be subdivided into subtasks where each sub-task stores results from preceding subtask that is needed for its computation. Data-dependent subtasks cannot run *concurrently* on multi-core processors due to their iterative nature. Zircon Software parallelizes these applications by interleaving multiple data-dependent subtasks to run in parallel. A case study trading strategy application using scenario analysis has been run on SGI Altix UV 1000 to show how one can use Zircon software to calibrate multiple trading strategies, each a data-dependent task, in parallel.

The results for the above two case studies show substantial performance improvement and scalability for both Heston calibration application containing data-independent tasks, and the trading strategies scenario analysis application containing data-dependent tasks. The Heston calibration application ran 34,400 financial models in 3 mins which is ~500 times faster than a serial run taking 24 hrs 24 mins on a 512-core SGI Altix UV 1000 system, thus showing near-linear scalability. With hyperthreading, the improvement was even ~40% higher, thus yielding ~666 times faster calibration time with 1024 threads. Similarly, the trading strategies analysis application ran ~479 times faster on a 512-core SGI Altix UV 1000 compared to a serial run and ran ~712 times faster on 1024 threads. These case studies show the ease with which Zircon software can be effectively used to parallelize data-independent and data-dependent applications on SGI Altix UV 1000 with minimal development effort and cost.

The remainder of this paper is organized as follows: Section 2 describes the Heston model calibration and scenario analysis applications; Section 3 discusses the limitations of applying conventional approaches for parallelizing applications; Section 4 describes the structure and functionality of Zircon software; Section 5 shows how Zircon software was used to parallelize the case study applications; Section 6 analyzes the results of experiments that quantify the performance improvements of the parallelized solutions; and Section 7 presents concluding remarks and lessons learnt.

2. Overview of the Case Study Applications

This section provides an overview of the *Heston Calibration application and Path-Dependent Trading Strategies Scenario Analysis application* that are used in this case study to demonstrate the parallelization capabilities of Zircon software.

2.1. Heston Model Calibration Application (Data Independent)

The first application in our case study is based on the work of Horn, Schneider, and Vilkov [7], who performed an extensive option pricing model calibration exercise to gauge the size and direction of the parameter misevaluation effect on hedging portfolio performance. The experiments reported in [7] processed thousands of individual Heston calibrations and took several days for the computations, which was far too long for typical research and practical purposes. It is essential to have calibration results for thousands of models within minutes or even seconds for industrial applications, such as risk management, hedging, or portfolio optimization. These calibrations run numerous times, especially for larger datasets, e.g., the original project reported in [7] used only five underlying securities for calibrations, whereas there are around 6,000 individual securities in OptionMetrics available for such analysis.

The Heston [6] model assumes the following risk-neutral dynamics for the underlying stock S and its local variance v:

$dS_t = rS_t dt + \sqrt{v_t} S_t dW^s{}_t$

$dv_t = \kappa(\theta - v_t)dt + \sigma_v\, v_t dW^v{}_t$

The parameters in the above equations represent the following:

- r is the risk free rate.
- θ is the long run variance mean; as $t \rightarrow \infty$, the expected value of $v_t \rightarrow \theta$.
- κ is the rate at which v_t reverts to θ, or speed of mean-reversion.
- σv is the volatility of the volatility, which determines the volatility of v_t.
- $E[dW_t^s\, dW_t^v] = \rho dt$, is the instantaneous correlation between the stock and the variance processes.

We used a non-linear least squares technique in our case study calibration to estimate five model parameters (starting variance value, long-run mean, speed of mean-reversion, correlation between the processes and the volatility of volatility) so that the theoretical prices get close (in terms of some norm) to the observed ones. The application uses an optimization routine that minimizes a 5-dimensional objective function (one dimension for each parameter in the Heston model). The objective function is calculated as the mean squared pricing error between the observed price and the calculated price for each combination of parameters.

We implemented the optimization routine us-ing the NAG nag_opt_bounds_no_derive() minimization function to run 100 iterations of the objective function and find the minima. The maximum number of iterations is capped to 100, so the model calibration is considered to have failed if the procedure does not converge by that point. We calibrate the Heston model using MID prices of available call options for OEX, with maturities ranging from 14 to 180 days and with moneyness (strike/stock price) in the range [80,120]. The observed prices are taken from OptionMetrics (www.optionmetrics.com), with the usual data-filters applied, e.g., we removed options with missing implied volatility, zero bid prices, and zero open interest. The theoretical option prices are calculated using the Fourier transform technique and involve some numerical integration.

2.2. Trading Strategies Scenario Analysis Application (Data Dependent)

The second application in our case study is based on scenario analysis, which is often used in automated algorithmic trading to calibrate the parameters of a particular model. The calibration process is based on optimizing the historically simulated performance of the strategy (with standard metrics like profitability, Sharpe Ratio, max drawdown, alpha, etc) as a function of unknown parameters. Although this exercise is not complex, it involves many computations that are often path-dependent. As a result, the change of position in some underlying instrument at any given point in time is a function of

- A signal from the option markets summarized by a deviation of an observed market price of an option from a theoretical option price computed using an option-pricing model for given (also observed) environmental variables and parameter values to be estimated, and
- The history of the past positions in the underlying instrument.

Individual strategy calibration cannot be generally parallelized for this application due to the data dependency between individual point computations for an instrument. We therefore scaled and accelerated this application to analyze up to 1024 trading strategies concurrently by leveraging all hyper-threaded 512 cores on the SGI Altix UV 1000 platform via Zircon software.

The application works as follows. First, we generate and save a large series of European call option prices with a fixed maturity and other parameters varying in a small range, which provides a historical series of observed option prices or the real-time stream of data. Second, we compute the position of an asset based on a signal from these observed call option prices. The current position in a given asset is denoted by θ^C for time t and stock j, and option type C and following algorithm is used to compute it:

1. At initial time, set $\theta^C = 0$.
2. At each time t =0…T, for each underlying stock j compute the theoretical value of the call option C^{th} using the binomial tree model with the input parameters equivalent to those of the currently observed call option from the saved series. The known input parameters from the series include stock price, strike price, interest rate, and expected dividends. The implied volatility parameter is drawn randomly for each strategy from a small range of values around the observed implied volatility.
3. Compare the computed theoretical value of an option $C^{th}_{j,t}$ with the observed option price and use past $\theta^C_{\tau,j}, \tau \in \{\max[(t-21,0),t\}]$ to determine new portfolio holding as follows:

(a). If abs $(C^{th}_{j,t} - C_{j,t}) < 0.05 * C^{th}_{j,t}, set\ \theta^C_{t,j} =$
$Mean\ (\theta^C_{\tau,j}),\ \tau \in \{\max(t-21,0),t\}$

(b). If $C^{th}_{j,t} - C_{j,t} \geq 0.05 * C^{th}_{j,t}\ , and\ Mean\ (\theta^C_{\tau,j}),$
$\tau \in \{\max(t-21,0),t\} < 1\ set\ \theta^C_{t,j} =$
$Mean\ (\theta^C_{\tau,j}), \tau \in \{\max(t-21,0),t\} + 1$

(c). If $C^{th}_{j,t} - C_{j,t} \leq$
$-0.05 * C^{th}_{j,t}\ , and\ Mean\ (\theta^C_{\tau,j}),\ \tau \in$
$\{\max(t-21,0),t\} > -1\ set\ \theta^C_{t,j} =$
$Mean\ (\theta^C_{\tau,j}), \tau \in \{\max(t-21,0),t\} - 1$

(d). Otherwise do nothing.

For each $\theta_{t,j}^{C}$ we can then easily compute the P&L of a generated strategy, and the respective performance metrics (we omit this step as non-essential for our analysis). As a parameter differentiating each strategy from the other, we use the implied volatility for the theoretical option price evaluation. In practice, this implied volatility parameter can come from an implied volatility surface calibrated under different assumptions. For our application, we randomly pick a value of implied volatility that is not far from the currently observed implied volatility for a given underlying stock.

3. Challenges of Parallel Application Development

Complex data analytics applications like Financials often rely on advanced analytical techniques such as market forecasting, portfolio analysis, trading, risk analysis, and option pricing. These applications typically process large amounts of data and have a constant need for acceleration in performance. One strategy for accelerating the performance of such applications is to use conventional high-performance computing (HPC) platforms, such as MPI [9], PVM [5, 8], OpenMP [11], CUDA [10], or Globus [4] for developing parallel implementations. These conventional HPC software platforms, however, have the following drawbacks:

- Price-per-core performance is not tied to linear gains in application speedups or compute processing. Conventional HPC software and hardware platforms are cost-prohibitive since they do not accelerate performance commensurately to the investment of resources allocated and do not adapt dynamically to changing workloads and resource availability. HPC systems are also typically expensive to build and maintain.
- Conventional HPC platforms are deployment-specific. They do not offer program once, run anywhere flexibility, requiring separate API and paradigms for colocated and distributed hardware that makes programming effort cumbersome and the resulting solution hard to configure and evolve.
- Conventional HPC platforms also require custom development and integration of custom server and application programming before they can work (and many common and legacy applications cannot be modified unless they are redeveloped).
- Conventional HPC platforms are often tied to modified applications. Once applications are customized, they are locked in to a particular HPC platform and deployment configuration, and cannot leverage updates without redoing the intense customization.
- Conventional HPC software platforms also require complex setup with no support for automated plug and play and customization to adjust the load manually on all processors in the network since they do not automate adaptive load balancing.

What is needed, therefore, is a solution that automatically deploys an *adaptive* ultra high-performance infrastructure across colocated and/or distributed processors with memory, maps relevant applications to pools of colocated and/or distributed processors and memory, manages their execution, and dynamically equalizes the workload in real time to maximally use available resources. Application developers can thus exploit the processing power and memory available to them, including newer technologies, such as multi-core processors, as well as traditional desktops and servers.

4. Solution Approach: Integrating Zircon Software on the SGI Altix UV 1000 Platform

Zircon software [2] provides C/C++ libraries[1] for parallelizing software applications, offers an adaptive ultra high-performance middleware platform that accelerates and scales up application performance, and addresses the challenges with existing strategies described in Section 3. Zircon software runs on all popular general-purpose and real-time operating systems since it is implemented atop the open-source ADAPTIVE Communication Environment (ACE) [12, 13], which shields applications from operating system dependencies without incurring the overhead of hypervisor-based virtualization mechanisms.

Applications built using Zircon software possess the following three characteristics:

- They are parallelized, i.e., their computations can run concurrently on multiple processors and/or cores with memory,
- They can be scaled elastically, i.e., the number of processors and/or cores with memory can expand or contract dynamically, and
- They are deployment agnostic, i.e., they can seamlessly deploy on a single multi-core processor or networked multi-core processors.

The process of applying Zircon technology is formalized by Zircon as a blueprint that pinpoints potential parallelism in customer software and identifies the best approach to exploit it on available hardware, ranging from colocated cores to interconnected clouds. This effort can be as easy as flattening existing loops or as complex as remodeling the software architecture to add new abstractions that parallelize critical program logic. To enable such adjustments, Zircon software provides power programming patterns and components that shield application developers from complex concurrency and networking mechanisms.

The Zircon software API supports both task and data parallelism patterns [3], which extend far beyond simple multi-core enablement of sequential software applications. For example, remodeling applications to run on colocated or distributed cores incur various challenges that have historically been addressed by unrelated and non-compatible APIs and tools. In particular, colocation strives to support short tasks, where potential network latency might rival the time needed to run a single task. As tasks get shorter, therefore, colocation must provide auto-batching and zero-copy techniques that balance workload among the cores. Distribution, on the other hand, makes application elastic, offering potentially unlimited scaling, while incurring inevitable latency costs, e.g., from data copy and network (de)marshaling.

Zircon software eliminates the trade-offs be-tween colocated vs. distributed task/data parallelism patterns via a common program. It has a run-anywhere API that enables software to run in deployments where the choice of colocation vs. distribution is hidden from client and server applications. Zircon software achieves ultra high performance via its use of custom binary protocols that incur less than 40 microsecond roundtrip average on typical 10Gb networks.

While application elasticity enables efficient use of hardware and massive workload parallelization, it cannot be harnessed effectively without support from the underlying service delivery platform. Zircon software provides highly efficient zNet™ service delivery platform to (1) intelligently equalize load, (2) identify routing destinations dynamically based on real-time feedback from distributed services, (3) recover missing requests, (4) tolerate network or service failure, (5) automatically discover services, and (6) monitor the health and status of application and system services.

SGI creates multi-core, cache-coherent scalable global shared memory systems by connecting several compute blades containing 2 of the Intel® Xeon® processor 7500 series, running at 2.26 Ghz, each with eight cores. These are eight core processors and use SGI's UV HUB ASIC [1]. These systems are well-suited for Zircon's colocation parallelism architecture. In this architecture, multiple worker threads can run in parallel to process large amounts of data in-memory.

5. Zircon Solution for Leveraging SGI Altix UV 1000 Platform

This section explains how we used the Zircon software described in Section 4 to parallelize, but not restricted to, the complex Heston model calibration and scenario analysis case study applications described in Section 2. We describe the sequential solution for each application first and then explain how Zircon's software is used to parallelize and scale the applications to run on 512 cores of the SGI Altix UV 1000 platform.

5.1. Zircon Solution for Heston Calibration Application

We accelerated the performance of the Heston calibration application to efficiently utilize 512 cores/1024 threads and large memory of the SGI Altix UV 1000 platform. As described in Section 2.1, the Heston calibration application uses an optimization routine that minimizes a 5-dimensional objective function (one dimension for each parameter in the Heston model). The objective function is calculated as the mean squared pricing error between the observed price and the calculated price for each combination of parameters.

We implemented the calibration in C++ using the Standard Template Library (STL) and the optimization routine was implemented via the NAG nag_opt_bounds_no_derive() minimization function to run 100 iterations of the objective function and find the minima. The maximum number of iterations is capped to 100, so the model calibration is considered to have failed if the procedure does not converge by that point. We calibrated the resulting model to the observed option prices by making 34,400 independent invocations to the optimization routine in a *for loop*, thereby calibrating 34,400 models.

5.2. Zircon Solution for Scenario Analysis Application for Path-dependent Trading Strategies

We also scaled the scenario analysis application to analyze up to 1024 trading strategies in parallel by leveraging 512 cores/1024 threads of the SGI Altix UV 1000 platform. As described in Section 2.2, the application computes portfolio position for each strategy at different time points ranging from time t = 0 to time t = 1000. These individual portfolio position computations for a strategy cannot be done in parallel because position calculation for θ^c at time t depends on θ^c value at time t–1. We therefore use the Zircon implementation of the application to leverage the SGI Altix UV 1000 system by concurrently analyzing 1024 trading strategies on 1024 worker threads for a given portfolio.

6. Analysis of Empirical Results

This section presents the results of experiments that quantify the benefits of parallelizing the original sequential implementations of the Heston calibration and the trading strategies scenario analysis applications to create the colocated implementations described in Section 4. Zircon solutions were run on an SGI Altix UV

1000 platform which contains 32 blades, with each blade containing containing 2 of the Intel® Xeon® processor 7500 series, running at 2.26 Ghz, each with eight cores (for a total of 512 cores) with hyper-threading enabled. The system had 4 terabytes of globally addressable shared memory, with each blade containing 128GB of memory, running under a single copy of the SUSE Linux Enterprise Server 11 SP1 operating system.

6.1. Results for the Heston Calibration Application

We measured the elapsed time for several runs of the Heston Calibration application, increasing the number of worker threads from 1 to 1024 on the hyper-threaded 512 core SGI Altix UV 1000 system. Figure 1a shows that the Zircon implementation of the Heston Calibration application takes 24 hours and 24 minutes to calibrate 34,400 models when run on a single core of the SGI Altix UV 1000 system. However, the application runs only in 3 mins on 512 physical cores showing a 500x improvement and a near-linear scalability, whereas, it runs only in 2 mins 11 secs while using 1024 threads, a further improvement of 40% due to hyperthreading. Figure 1b shows ~666 times faster calibration time with 1024 threads.

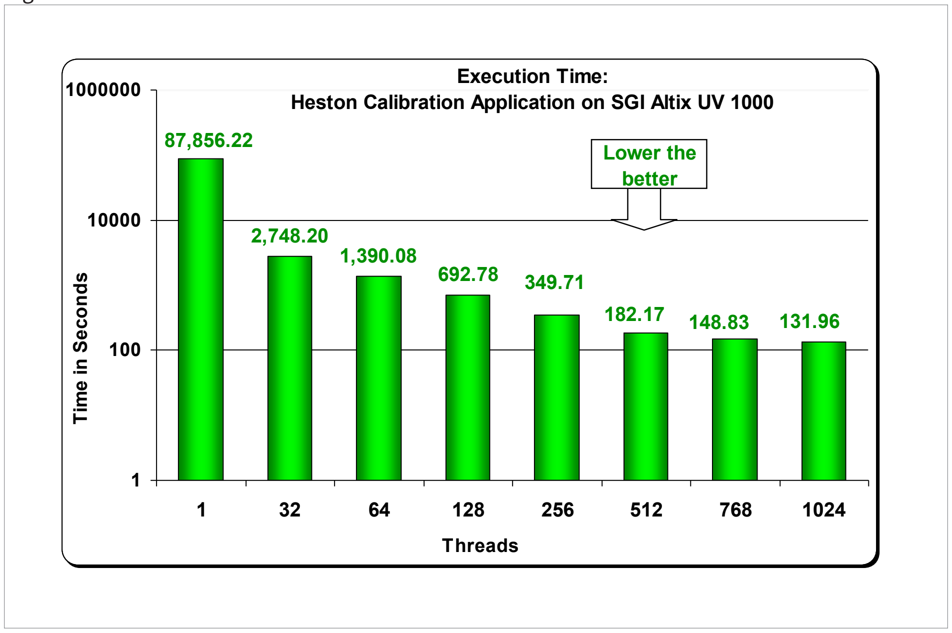

Figure 1a: Execution Time: Heston Calibration Application

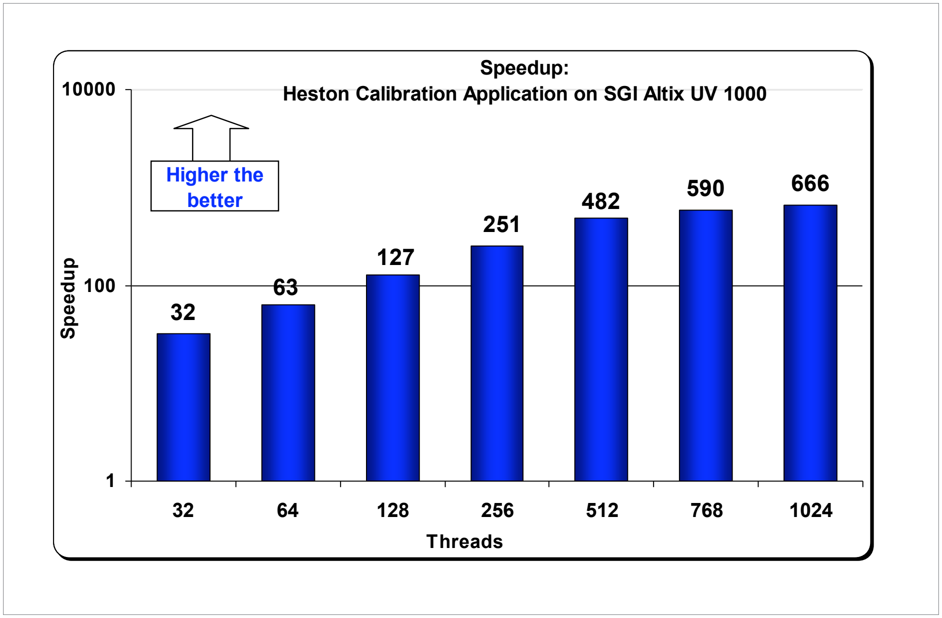

Figure 1b: Speedup: Heston Calibration Application

Results show that the Zircon implementation of the Heston calibration application scales linearly on SGI Altix UV 1000 to calibrate more models/second as the number of threads in the application are increased.

6.2. Result for Path-dependent Trading Strategies Analysis Application

Figures 2a, 2b, 3a and 3b show the results for path-dependent trading strategies analysis application. In the first run of this experiment, we analyze 1, 4, 8,16, 32, 64, 128, 256, 512, 768 and 1024 trading strategies in parallel by running 1, 4, 8, 16, 32, 64, 128, 256, 512 , 768 and 1024 threads, respectively, where each strategy computes θ for *1,000 time points* and thus runs *1,000 iterations each*.

Each individual iteration of θ computation ran for *104 milliseconds*. The results in Figure 2a show that the number of θ computation iterations per second goes up with the number of threads, thereby yielding a higher throughput. Figure 2b shows a *479 times speedup* on a 512-core SGI Altix UV 1000 system compared to a serial run and a *712 times speedup* on 1024 threads.

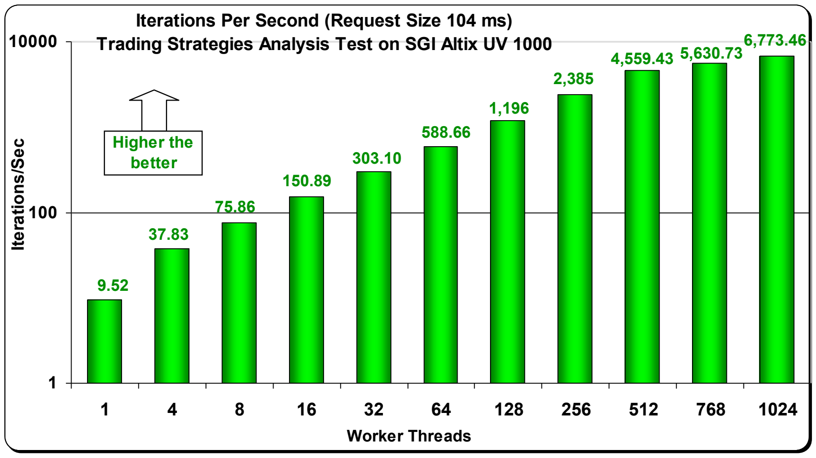

Figure 2a: Iterations Per Sec: Trading Strategy Analysis Application (104 ms/iteration)

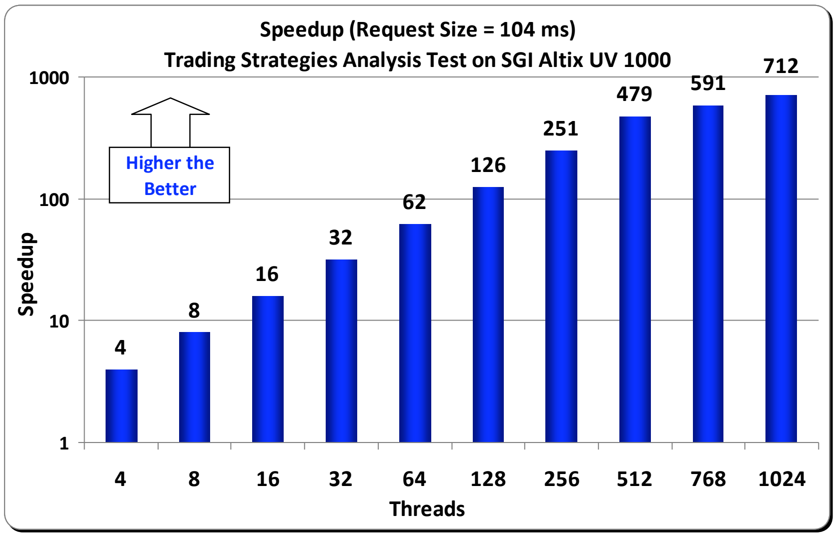

Figure 2b: Speedup: Trading Strategy Analysis Application (104 ms/iteration)

In the second run of this experiment, we analyze 1, 4, 8,16, 32, 64, 128, 256, 512, 768 and 1024 trading strategies in parallel by running 1, 4, 8,16, 32, 64, 128, 256, 512 , 768 and 1024 threads, where each strategy computes θ for *42,000 time points* and thus runs *42,000 iterations each*. The time taken by each iteration in this experiment is *2.5 milliseconds*, so we use the batching capability of Zircon software to perform 42 iterations simultaneously to overcome operating system scheduling overhead. The results show the number of iterations computed per second goes up with the number of threads, thereby yielding higher throughput. Figure 3a shows the throughput to increase with threads, for this second run of path dependent trading strategies analysis application. Figure 3b shows ~*400 times speedup* on 512 cores and an additional 14% improvement with hyperthreading.

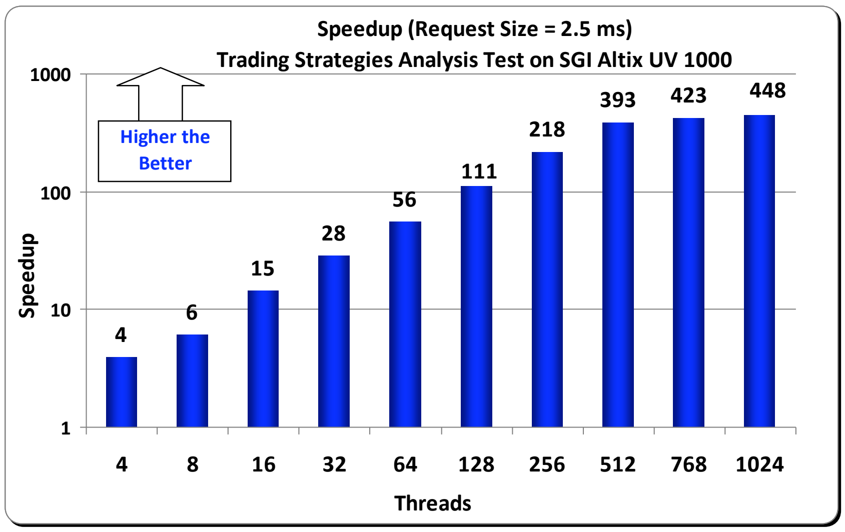

Figure 3a: Iterations Per Sec: Trading Strategy Analysis Application (2.5 ms/iteration)

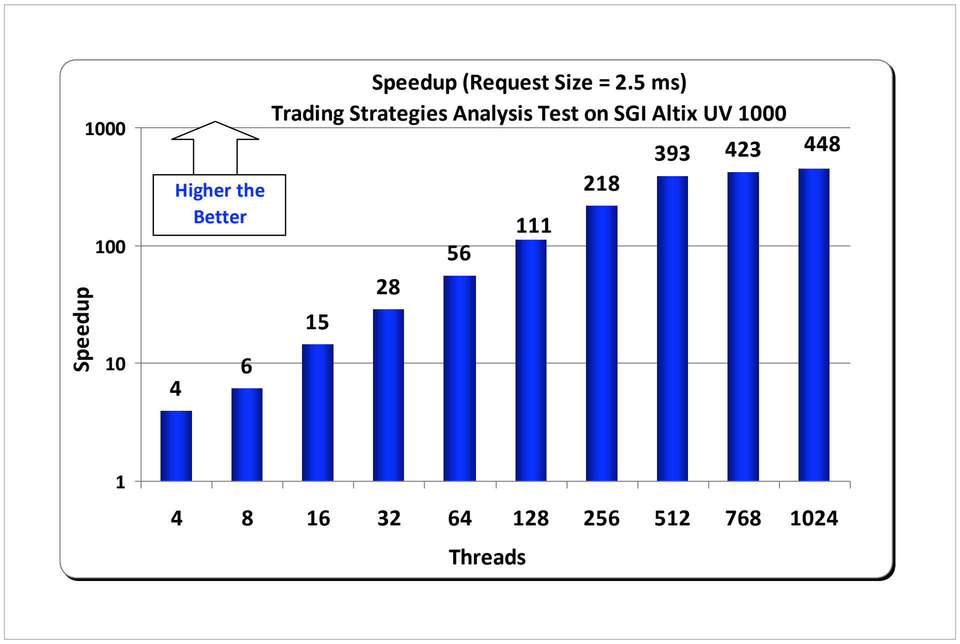

Figure 3b: Speedup: Trading Strategy Analysis Application (2.5 ms/iteration)

7. Market Focus

The market focus for the combined solution of Zircon Adaptive Software on SGI Altix UV 1000, are as follows:

- **Financial Services**
  - Risk assessment and modeling;
  - High Frequency Trading;
  - Real Time Analytics based on algorithmic feedback.
- **Government**
  - Detecting Security Threats;
  - Analyze volatile markets;
  - Weather Forecasts.
- **Healthcare**
  - Accelerate processing of huge image files such as MRIs;
  - Complex analyses of patient diagnoses, treatments and outcomes to identify risk factors.
- **Digital Content Management**
  - Processing large graphical images (medical MRI, video animation);
  - Processing, archiving, storing and searching individual documents and content repositories for enterprise content management systems.

8. Conclusions

This paper demonstrates how the Zircon soft-ware can be used to leverage a massively parallel, large memory system such as SGI Altix UV 1000 with 512 cores, 4 TB memory to accelerate complex data analytics applications in a near-linear scalable manner. The case studies with two Financial Applications show how Zircon implementations of applications can significantly accelerate performance of their corresponding sequential implementations on multi-core, large memory architecture of SGI Altix UV 1000. As shown in Section 6, Zircon software improved performance dramatically, with minimal learning curve and configuration/deployment effort.

The following are the advantages in developing and zEnabling applications using Zircon software on the SGI Altix UV 1000 platform:

- Zircon Adaptive Software is easy to learn and application developers are shielded from a tedious and error-prone low-level shared memory and parallel programming exercise. Zircon software allows programmers to focus on what they do the best—focus on the business logic—and leave the complex details related to shared memory programming to Zircon infrastructure.
- Acceleration benefits for data analytics applications using Zircon software cannot be attributed solely to the multi-core architecture of SGI Altix UV 1000 but also to its consolidated and glued architecture, its capability to combine processors and memory and to run a single image of operating system. This results in an extreme scale-up capability on SGI Altix UV 1000 for applications like Financials; Digital Content Management; Security threat and Fraud Detection, Market risk analytics by Government; as well as MRI processing and identifying patient's risk factors in the Healthcare space.
- On a flexible system like SGI Altix UV 1000, data analytics applications can be quickly parallelized with variant core-levels and memory with a predictable performance and native real-time load equalization and near-linear scalability.

Participants

Zircon Computing. Zircon Computing, LLC, is an international software and services company based in Wayne, New Jersey. Founded in 2005 by senior technologists from the financial services industry, Zircon Computing is a leading provider of ultra high-performance middleware software and services worldwide, and markets both directly to enterprise clients and through an international network of partners. Zircon Computing is privately held. For more information, please visit http://www.zircomp.com.

SGI. SGI (NYSE: SGI) is a global leader in large-scale clustered computing, high performance storage, HPC and data center enablement and services. SGI is focused on helping customers solve their most demanding business and technology challenges. Visit http://www.sgi.com for more information.

References

[1] Technical advances in the SGI Altix UV architecture. http://www.sgi.com/pdfs/4192.pdf.

[2] J. Balasubramanian, A. Mintz, A. Kaplan, G. Vilkov, A. Gleyzer, A. Kaplan, R. Guida, P. Varshneya, and D. C. Schmidt. Adaptive Parallel Computing for Large-scale Distributed and Parallel Applications. In Proceedings of the 1st International Workshop on Data Dissemination for Large scale Complex Critical Infrastructures (DD4LCCI 2010), Valencia, Spain, Apr. 2010.

[3] F. Buschmann, K. Henney, and D. C. Schmidt. Pat-tern-Oriented Software Architecture: A Pattern Language for Distributed Computing, Volume 4. Wiley and Sons, New York, 2007.

[4] I. T. Foster. Globus toolkit version 4: Software for service-oriented systems. J. Comput. Sci. Technol., 21(4):513–520, 2006.

[5] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. PVM: Parallel Virtual Machine A Users' Guide and Tutorial for Networked Parallel Computing. MIT Press, 1994.

[6] S. Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. Review of Financial Studies, 6:327–343, 1993.

[7] D. Horn, E. Schneider, and G. Vilkov. Hedging options in the presence of microstructural noise. SSRN eLibrary, 2007.

[8] D. Kranzmuller, P. Kaczuk, and J. Dongarra. Recent advances in parallel virtual machine and message passing interface. IJHPCA, 19(2):99–101, 2005.

[9] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard, Version 2.2. High Performance Computing Center Stuttgart (HLRS), September 2009.

[10] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming with cuda. Queue,6(2):40–53, 2008.

[11] OpenMP Architecture Review Board. OpenMP application program interface. Specification, 2008.

[12] D. C. Schmidt and S. D. Huston. C++ Network Programming, Volume 1: Mastering Complexity with ACE and Patterns. Addison-Wesley, Boston, 2002.

[13] D. C. Schmidt and S. D. Huston. C++ Network Programming, Volume 2: Systematic Reuse with ACE and Frameworks. Addison-Wesley, Reading, Massachusetts, 2002.

ZIRCON COMPUTING

intel Xeon inside

Powerful. Intelligent.

Corporate Office
46600 Landing Parkway
Fremont, CA 94538
tel 510.933.8300
fax 408.321.0293
www.sgi.com

North America +1 800.800.7441
Latin America +55 11.5185.2860
Europe +44 118.912.7500
Asia Pacific +61 2.9448.1463