



i n v e n t

**SCIENTIFIC**  
COMPUTING ASSOCIATES  
INC.

## **A Benchmark Report using parallelR™ for High Performance Portfolio Optimization on HP Clusters**

### **1. Introduction**

Virtually all Financial Services enterprises are facing increasing levels of competition in the global economy that has tended to level the competitive playing field. Enterprises that depend on the web as a primary interface with their constituencies have a particularly acute problem. Not only do they have to collect, digest, and analyze ever increasing amount of data, but they also must respond with answers and action in real-time to be effective and remain competitive.

It is widely recognized that the business opportunity and challenge is to develop and utilize modern high performance computing technology to derive rapid response, adaptive, robust Financial Services to customer. High performance computing is playing an increasingly important role in the Financial Services industry

Meeting this challenge is difficult and expensive. The conventional “scale up” approach utilizes proprietary Shared Memory Multiprocessors. However, such systems are relatively expensive and improvements in processor speed haven’t live up to Moore’s Law in the past four years [3]. In contrast, scientists and engineers have been successfully utilizing a “scale out” strategy of parallel computing on commodity Beowulf clusters. These systems utilize inexpensive commodity hardware, but the development of suitable “scale out” parallel application software can be very difficult for most programmers.

The cost effective development of parallel Financial Service applications that can leverage commodity clusters to provide required performance is a key enabler of the scale out strategy. Fortunately a large number of the statistical and optimization components of Financial Services are “embarrassingly parallel” and can be easily programmed using newly available software tools.

In order to keep software development and maintenance costs down, application developers are increasingly using high level scripting languages for software development as an alternative to traditional compiled languages such as Fortran or C. For purposes of this study we have employed the well-known interactive R statistical computing environment, cf. [1]. We assume the reader has some familiarity with R. We utilize the parallelR package from Scientific Computing Associates Inc.[2] to parallelize a simple prototype portfolio optimization package.

By using the Sleigh function in parallelR, an R user can quickly develop and run embarrassingly parallel Financial Services applications with very good results without becoming trained in parallel programming. We benchmarked the scale-out capability of the parallelR package on synthetic market data for the purpose of producing an easy to understand illustration of the general concept.



i n v e n t

**SCIENTIFIC**  
COMPUTING ASSOCIATES  
INC.

Our results show that the system provides excellent performance and scales very well with increasing numbers of CPU cores. Perhaps as important, we demonstrate that by utilizing the parallelR system, this type of parallel application software can be developed and run by a typical R user who is not an expert in parallel programming. Indeed the parallelR system has been designed and implemented with the goal of minimizing the incremental “cognitive load” that a typical R user need to face in utilizing parallel computing. This is a critical observation because conventional approaches to parallel application development are notoriously difficult and time consuming.

## 2. The Problem

Opportunities for utilizing high performance Financial Services systems are common throughout the financial world. Examples include: credit risk assessment, portfolio optimization, optimization of marketing strategies, and credit card fraud detection.

The specific example we consider is a simple, prototype portfolio optimization problem that is a model of the “efficient frontier” approach suggested by Markowitz. cf., [4]. Our intention here is to illustrate the general ideas behind the use of HP clusters to accelerate general portfolio optimization rather than to present the design and analysis of a production portfolio optimizer applied to real market data. We present benchmark data showing how well our parallel portfolio optimizer scales as a function of the number of cores.

Our model portfolio consists of fixed number of investments whose prices are characterized by randomly selected normal distributions. We want to emphasize that in the real world of finance, an enormous amount of (proprietary) science and art goes into designing the distribution functions that model the behavior of individual investments. We use random normal distribution functions to represent investments because our entire focus here is on the parallel computing aspects of this problem.

We use a Monte Carlo approach to evaluate the reward/risk of a portfolio chosen from these investments with only “long positions” being allowed, ie., no investments are allowed to be “sold short.” Monte Carlo approaches to studying efficient frontiers are of more than academic interest. Schlumberger has been using similar approaches with great success for analyzing reward/risk models in the petroleum with great success for years, cf., [4].

Each investment has an associated weight,  $w$ , which is the nonnegative fraction of the total portfolio value that is allocated to that investment. Clearly, each weight,  $w$ , must be a fraction between 0 and 1.

Specifically, we generate a large number of random portfolios of “long positions” and for each such portfolio we compute the expected reward and risk by another Monte Carlo process. To be precise, we randomly sample each investment in the portfolio a large number of times and compute the mean (average reward) and risk (variation) for the portfolio. The resulting (reward, risk) points can be plotted in a plane to yield a collection of points. The upper bound of the convex hull of these points is generally called the “efficient frontier.”



### 3. Discussion

Portfolio optimization problems can be very compute intensive and in practice variants may need to calculate in real time because of market considerations.

This sequential computation is simple to implement in R and the parallel version is naturally expressed in the Sleight function from SCAI's parallelR [2]. The program that studies this problem is remarkably short which illustrates the power of R as a statistical modeling languages and parallelR as a parallel programming language.

Specifically we parallelized the outer loop of our simple code in which the random portfolios are constructed. In theory this can create a large number of tasks, each of which involves the evaluation of the reward and risk of a random portfolio (or a subset of them). In our model this evaluation is itself calculated from the given distribution functions by means of Monte Carlo procedure.

### 4. The Computer Platform

Our benchmark was run on a six node HP cluster. Each node was equipped with dual, dual-core AMD Opteron™ 875 chips. The clock speed on the chips was set at 1.8GHz. Four out of six nodes had 4GB RAM, and remaining two had 2GB RAM. Each node was running Red Hat Enterprise Linux AS release 4, R 2.3.1 built with gcc and g77 3.4.4. and parallelR 1.3.3.

### 5. Benchmark Results

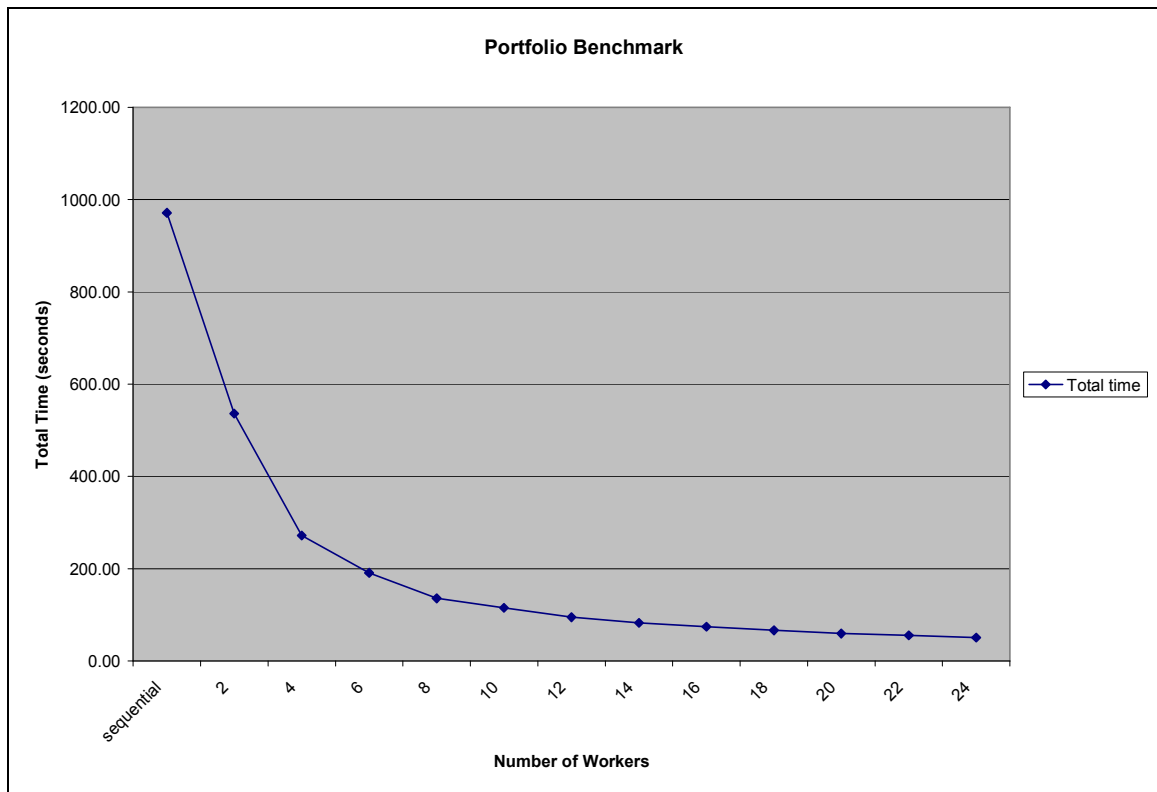
The benchmark we studied consists of 3600 portfolios and each of which contained 25 stocks. To calculate the reward and risk for each stock, we used a Monte Carlo approach. We randomly sampled 10000 points from each stock's distribution function. We then took the inner product of the stocks and their weights. This gave us the reward, which was used to calculate average reward and risk (variance of the reward). Figure 1 shows the amount of time it took to analyze 3600 portfolios with a varying number of Sleight workers. Figure 2 shows the speedup of different number of Sleight workers compares to sequential version program. There is a factor of 3.6 speedup with four Sleight workers, a factor of 7 speedup with eight Sleight workers, and a factor of 19 speedup with 24 Sleight workers.

The following chart reflects the workload timings per the # of workers applied to the portfolio problem. As depicted in the matrix, the parallel activity represents an almost linear speedup when adding workers to the problem. This is further illustrated in the attached Figures 1 and 2, which graphs the workload against the number of workers applied to the parallel execution and load balancing applied when using Parallel R technology. We are very satisfied with the linear speedup attributed to the HP Cluster interconnect using simple Gigabit Ethernet and the overall performance of the AMD Opteron Dual Core Blade technology, designed by HP in their ProLiant BL45 Blade Servers.

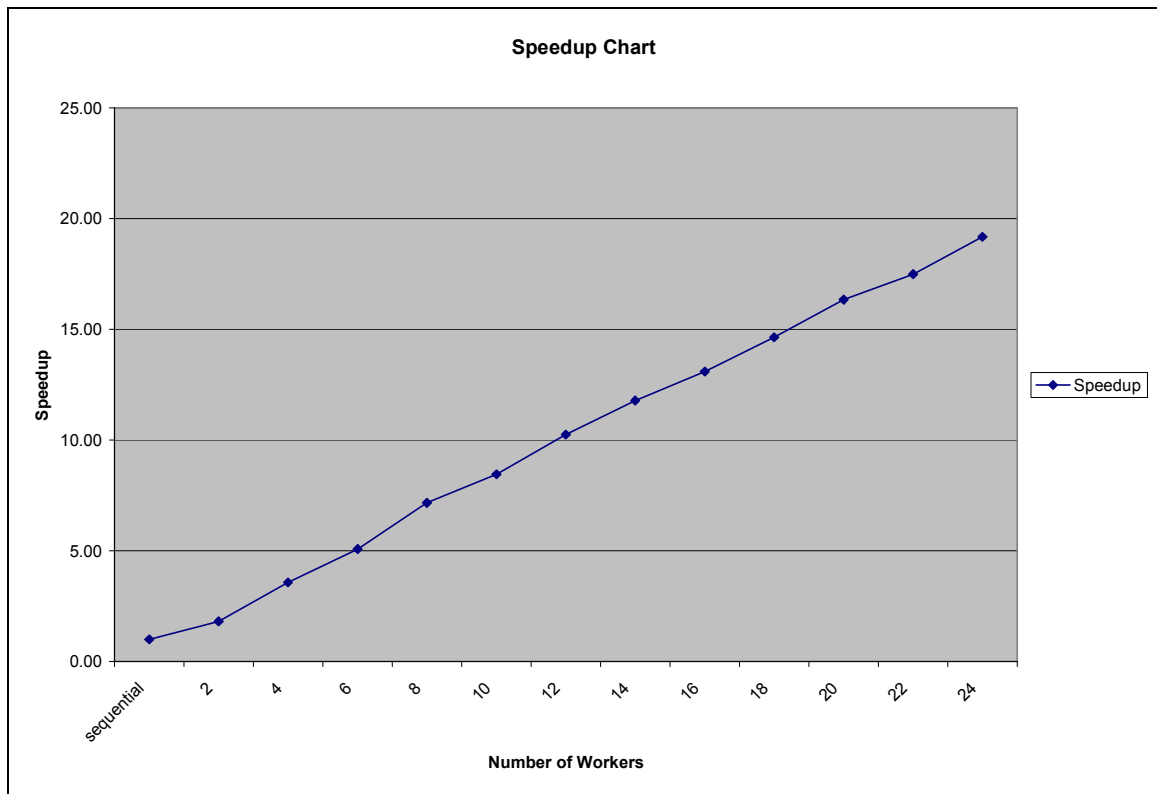


# of workers	Initialize time (seconds)	computation time (seconds)	Total time (init + comp)	Speedup (sequential divide by # of workers)
sequential	0.00	971.49	971.49	1.00
2	0.01	536.23	536.24	1.81
4	0.01	272.08	272.09	3.57
6	0.01	191.22	191.23	5.08
8	0.01	135.60	135.62	7.16
10	0.01	114.94	114.96	8.45
12	0.02	94.77	94.79	10.25
14	0.02	82.38	82.40	11.79
16	0.03	74.14	74.17	13.10
18	0.03	66.31	66.34	14.64
20	0.04	59.42	59.45	16.34
22	0.04	55.47	55.51	17.50
24	0.04	50.61	50.65	19.18

**Table 1. Benchmark result**



**Figure 1. Amount of time versus the number of workers to finish a portfolio analysis.**



**Figure 2. Amount of speedup versus the number of workers.**

## 6. Conclusions

We draw three main conclusions from our study:

1. For the parallel decision support application that we studied, the dual-core AMD64 based HP Cluster provides remarkable scalability up to the natural limits of the problem size.
2. By using the `parallelR` package, an R user can easily create and run parallel R programs that make efficient use of cluster hardware.
3. HP clusters with appropriate programming tools promote the quick and cost effective solution of problem. Jobs are completed in a timely fashion.

## 7. References

[1] [www.r-project.org](http://www.r-project.org)

[2] [www.lindaspaces.com](http://www.lindaspaces.com)



[3] [http://www.tomshardware.com/2005/11/21/the\\_mother\\_of\\_all\\_cpu\\_charts\\_2005/](http://www.tomshardware.com/2005/11/21/the_mother_of_all_cpu_charts_2005/)

[4] Monte Carlo: An Alternate Approach to Efficient Frontier  
Balancing portfolio risk and return with efficient frontier  
By: Jason McVean.

[http://www.slb.com/content/services/software/valuerisk/expert\\_paper\\_monte\\_carlo.asp?seg=www.pipesim.com&printView=true&](http://www.slb.com/content/services/software/valuerisk/expert_paper_monte_carlo.asp?seg=www.pipesim.com&printView=true&)

**Benchmark Configuration:**

Qty	SKU	Scientific Computing Associates Benchmark Configuration Description
		<b>Blade Enclosure</b>
2	243564-B22	HP BLp Enhanced Enclosure
2	283192-B21	HP BLp-Class C-GbE2 Interconnect Switch
		<b>Blade Configuration</b>
6	397814-B21	HP BL45p 2.8GHz SC 2G 2P Svr
18	376639-B21	HP 2GB Reg PC3200 2x1GB Memory
6	286714-B22	HP 72GB 10K U320 Pluggable Hard Drive
		<b>Rack, PDU's, Monitor</b>
1	245163-B22	HP 10622 22U Shock Rack Pallet
1	221546-001	HP TFT5600RKM Integ 1U US KeyBrd&Monitor
2	252663-B24	HP 16A High Voltage Modular PDU
		<b>OS and Applications</b>
1	366320-B21	SUSE SLES9 1Yr Base 8pk ET/Opt
2	399712-B21	SCAI Prlel RI Pro 1st yr Sub/Support- 4 Node
		<b>Total Hardware and OS \$99,222.00</b>