



## Grid on the job

Using grid computing for pension calculations makes huge business sense.

By **Julie Bort, Network World**

When network executives think of grid computing, they often envision the supercomputer-strength computational resources needed for academic or government research projects. That's certainly how IT leaders at human-resources outsourcing and consulting firm Hewitt Associates once saw grid.

About 18 months ago, while working on on-demand computing and other advanced technology initiatives, Dan Kaberon, Hewitt's director of computer resource management, told IBM that no business applications existed for grid. But then he got to thinking about one particularly nasty IT-related business problem the company was coping with, and changed his mind. So after being a vocal naysayer of using a grid for business applications, "it was then up to me to find one," he says, with a laugh.

Working in partnership with Tim Hilgenberg, Hewitt's chief technology strategist for application development, Kaberon created a Linux blade server grid, ported portions of a troublesome mainframe application to it and reduced transaction costs by 90 per cent. In the process, Hewitt demonstrated that grids can solve pressing business needs.

At issue was a mainframe-hosted application that performed complex pension calculations for 5.5 million of Hewitt's customers, many of whom work for multinational corporations that have grown through acquisitions.

The pension application, written in Smalltalk running under Customer Information Control System (CICS) and DB2, uses input data such as length of employment, vested percentages, pension payout requirements (which grow complicated after acquisitions and reorganisations), performance of investments and other statistics. It then crunches those numbers to produce estimated monthly pension payments. Users also can perform what-if analysis to model various retirement options.

The application was very popular, but use would fluctuate wildly. Whenever rumours of workforce actions such as mergers, acquisitions or early retirement programs circulated, thousands of employees would go online at the same time to perform pension calculations, Hilgenberg says.

Obviously, Hewitt had no way of knowing when such murmurings would happen. "The application consumed over 1,800 mainframe MIPS and volume could double without warning," Kaberon says. Those big swings of volume "would consume huge amounts of computer power, and, at that time, all of that on a mainframe, a very expensive place to run something as mathematically intensive as pension calculations," he says.

The application made an ideal grid candidate for another reason, too. Beyond a quick database query to collect the data needed to perform its calculation, the application required little interaction with other IT systems and data. "You could take your knife and cut out the application from the rest of everything without causing much complexity," Kaberon says.

### Building a grid

With a grid, Kaberon and Hilgenberg hoped to off-load the pension calculator's number-crunching from the mainframe onto Intel hardware, which is cheaper to buy, easier to maintain, fix, replace and manage, and needs less-expensive software. So they asked IBM to help build a grid proof of concept for the pension calculation application. That first design "demonstrated we could build a successful production environment out of it," Kaberon says.

The two IT executives then gathered about 20 people from their respective staffs, plus engineers from IBM and grid middleware vendor DataSynapse, and formed a grid project team. Within three months, the team had built the pension calculation engine grid, which has been running successfully in production since September 2003.

The grid, which essentially acts as extra CPU power for the mainframe, was initially built with 10 IBM Linux eServer blade servers divided into two mirrored configurations placed at each of Hewitt's two Lincolnshire data centres, with a

variable number of servers in use at any given time. The Linux blade servers run DataSynapse GridServer middleware, and Java-based middleware connects the grid to the calculation application on the IBM mainframe.

Here's how it works. A user fires up a browser and logs onto the Hewitt-hosted "Your Benefits Resources" site. On the pension information page, the user clicks on a link to calculate a monthly pension payment. That link initiates a call to a Web application running on a Sun Solaris server, which makes a call to the mainframe where the plan administration applications run. The mainframe-based pension calculation application collects necessary information from databases and, via a Web services connection, turns that information over to the GridServer middleware for dispatching of the number-crunching to the CPUs in the grid. The grid behaves like a giant CPU, calculating the numbers and returning the result to the mainframe. That result winds its way back through the Web services connections to the user's browser. If a user wants to play with the numbers - retirement next year instead of next month - the process starts afresh.

When factoring in the Intel-based hardware, Linux, the grid middleware and ongoing support, the cost of performing a pension calculation on the grid is 10 per cent of what it had been on the mainframe, Kaberon says.

### **From one app to another**

With such impressive savings and a better understanding of what constitutes a grid-worthy application, Kaberon and Hilgenberg soon found another IT problem where their months-old grid could come to the rescue. In this case, the problem involved an application that performed a printing composition job. Whereas the pension calculator shuttles a high volume of small but compute-intensive requests over to the grid to execute, the print composition application will break one extremely large computational workload into small parts, then use the parallelism the grid affords to complete the work faster, Hilgenberg says. This second grid application is now in early production.

Hewitt customers use the print composition application prior to benefit open enrolments. "Imagine preparing eight to 12 pages of [Adobe Postscript or PDF] worksheets for every one of a 100,000-employee company explaining service credits, health and general benefits options by employee role, seniority and region. At the end of enrolment, each of these employees also may get an enrolment confirmation statement," Kaberon says.

On the mainframe, composing all those forms might take 100 hours to complete, with the processing spread over a week or more as the mainframe shoulders this huge computational task in addition to its usual workload. If the print computational task hung during hour 99, the whole of it would need to be started from scratch.

Enter the grid. The bottleneck of the print composition application was the enormous amount of computation required. By porting the print rules to the grid and breaking the task into 100 processes that each required one hour of compute time, the grid could finish those tasks simultaneously. A process that used to take 100 hours could be done in an hour.

Expertise with the first grid application had shown Kaberon that the grid has a multitude of uses that only require a little creativity to recognise.

"The working hypothesis that I presented to the CIO and various technical groups was that if we build good competence around grids, then we would begin to recognise other opportunities that were good grid candidates that we just couldn't understand. I had absolutely no idea what those might be," Kaberon says. As it worked out, "the composed print application just stuck right out and said, 'Here's one.' "