Building Performance into the Development Process

Bruce Schneider, Cray Research, Inc., System Performance and Analysis Section, Eagan, Minnesota

Introduction

The System Performance and Analysis Section (SPA) in the Operating System Group has changed it's direction over the last year to focus on evaluating performance up-front instead of after the fact. This section previously reported on performance, now it impacts performance during the development process. An example of this is at the Spring CUG in 1994 a member of this section gave a talk on the performance of the UNICOS 8.0 operating system. While this was interesting to most of you, some of you had already installed and were running 8.0. The performance results gathered were after the software was designed and implemented. This year, I will talk on Friday about performance issues of a modular distributed operating system (UNICOS/mk) on the upcoming T3E architecture. Instead of talking about performance of a product already out, I will talk about the performance of a system under development; a system that we are building performance into. To give you background first though, I have to talk about how we build performance into the development process.

This change in philosophy from evaluating systems to providing input into the development cycle has many driving factors. The main factors, which is affecting all industries, is time to market and competition. Getting the right product out to market in a timely fashion is crucial to surviving. To address this, CRI is creating shorter hardware development cycles which in turn shortens the software development cycle.

In order to address the pressures of time to market and shorter development cycles, the time and cost of software design be reduced. Redevelopment due to inaccurate designs must be eliminated. Software designs must: be developed quicker, be right the first time and perform optimally. The ultimate goal being the ability to provide well thought out performance based software quickly to our customers. The only way to do this is to build performance into the development process.

One key component to building performance into a system is knowing customer workload characterization. Workload characterization is the knowledge, data, or sample of some specific sequence of events. In designing the performance into a piece of software, the closer the data used to test the software mimics what a customer will run in a production environment, the better the end design will be to meet actual customer requirements.

Workload characterization data is used as input data to drive models. This makes models more predictive of real customer workloads. The second use of workload characterization is that the workload becomes a basis for actual performance tests.

An example of customer workload characterization is we polled data from ten UNICOS sites representing diverse industries and classified the system call usage. From this we created a synthetic workload that mirrored the percentage of system call usage we gathered. We made these tests reproducible and predictable and use this for performance analysis to look for code regressions in UNICOS releases.

We have experience at knowing what customers run on a PVP machine but we believe this will not be the same type of workload on an MPP. We are in the process of gathering customer workload characterizations for the MPP to create input data to drive our MPP models.

The first phase of building performance into a system is modeling. Modeling allows early software access to new hardware. By modeling the characteristics of the hardware, various software design alternatives can be explored. From this, expected performance levels of the software designs can be determined. This feedback early in the development cycle allows for design decisions to be made based upon performance characteristics. Because modeling uses an analytical technique rather than relying on intuition, design decisions can be made with relative confidence.

An example of a model we created was to answer the question of the I/O capability of Serverized UNICOS on the T3E architecture. When the UNICOS operating system was multithreaded we had no analytical data to show the expected improvement until after the code was developed and actual tests run. With modeling we can predict performance results before major code development decisions are made.

Three modeling projects related to: T3E architecture, Serverized UNICOS and the SCX channel are detailed in the talk titled "Modeling Serverized UNICOS on the T3E Architecture".

The next phase of building performance into a system can be simulation. Simulation can provide a finer level of granularity than modeling and also validate the modeling results. Simulation allows you to emulate software running on new hardware. An example of simulation that we currently use is a T3E hardware instruction set simulator that allows us to develop and verify T3E software such as the operating system and compilers.

The last phase of building performance into a system is verification; the testing of the software on the actual hardware. Tests can be written to exercise the software on the hardware and determine timing results. These results can then be compared back to the predicted modeling and simulation results. This validation is a loop whereby the actual testing results verifies the accuracy of the modeling and simulation. This feedback loop allows for the knowledge to create more accurate models and simulations on future projects. Modeling and simulation may have pointed out areas of concern that should be exercised thoroughly by creating specific tests. These performance tests can then become the basis for future regression testing.

Another key to testing is to perform regression testing such that it is timely enough to have an impact upon the development process. Currently we run our performance workload tests on a weekly basis. In a sense this is continuous testing. The idea is to find any performance regression and identify the specific code within two weeks of code integration while the code is still fresh in the developers mind. An example is while running the customer system call synthetic workload test we noted a regression in some non-shared File System I/O calls from code being developed for the Shared File System. From the time the code was integrated into the system, a regression was determined, a piece of code identified as causing the regression and the appropriate developer notified, took less than 5 days.

Summary

Cray Research customers use models, simulation and observation to analyze everything from financial trends to engine combustion, even the weather. I believe if you can use these tools to predict something as unpredictable as the weather we should be able to model software design decisions. The key is though while you may not be able to change the weather, we believe we can impact software and hardware performance.