Performance Monitoring

A Report for the NOTUR Project *Emerging Technologies: Cluster*

Lars Ailo Bongo, Otto J. Anshus, John Markus Bjørndalen
Department of Computer Science, University of Tromsø

In this note we present our experiences when evaluating various performance monitoring tools for use in a study where the goal was to examine if and how the communication behavior of various benchmark applications changes when the problem size is scaled down.

As our research is focused on scalable communication systems for clusters, we need benchmarks that stress the communication system, i.e. that spend most of its time communicating. One way to achieve this is, to scale down the problem size, and thus increase the communication-to-computation ratio.

The approach chosen for the study is to use various performance monitoring tools to analyze the changes in the communication and computation behavior when the problem size is scaled down.

We are especially interested in monitoring the usage of the communication system. All the performance tools mentioned in this document are described in our survey ‘‘Survey of Execution Monitoring Tools for Computer Clusters’’, done for the NOTUR project Emerging Technologies Cluster.

The benchmark applications used are written using MPI. This allow us to evaluate at least two different communication systems; LAM/MPI and MPICH. However, many of the MPI monitoring tools did only support one of the MPI implementations (even if the MPI standard defines a profiling interface). Paradyn and Vampir 3.0 only supports MPICH, and XMPI only supports LAM-MPI (however, we never actually used XMPI with LAM-MPI).

At the time of the evaluation Vampir 4.0 did not support LAM-MPI., and XMPI did not support the latest version (7.0) of LAM-MPI.

The tool bundled with MPICH, Jumpshot, required a recompilation of the MPICH library.

Based on the experiences gained from this work, our conclusion is that installing and using performance monitoring is a lot of work. Also many of the monitoring tools are not very portable.

The second part of the evaluation was to install tools for monitoring the computation behavior. The tools evaluated all use the performance counters on Intel Pentium processors because we had readily available a local Intel-based cluster.

In practise, this turned out to be even more time cnsuming than installing the MPI monitoring tools. A kernel patch was required for PAPI, while VTune required a specific version of the RedHat Linux kernel. As our clusters runs a rather old Linux kernel none of the tools could be installed.

The PAPI module will be installed when we update the operating system of the clusters.

The conclusion is that care must be taken to make sure the tools will work for a given release of the operating system.

We used the Vampir tool to evaluate the embarrasingly parallel Monte Carlo Pi application. As we only had an evaluation version license on Vampir we only had time to examine the Monte Carlo Pi application. As we did not have a Vampire license available, or could find one in use elsewhere inside the project which we could use, we contacted Pallas, the company that has developed Vampir. Pallas provides us with a temporary license to be used in the project described in this note.

Vampir provides many useful visualizations that allowed us to conclude that the communication behavior of the application does change as the problem size is scaled down. However, to our knowledge, no information is provided by Vampir on the performance of the communication system. Such information would require Vampir to have knowledge about the architecture and implementation of the system. This is not surprising as Vampir and most other monitoring tool for parallel applications are targeted at monitoring the parallel applications and not the underlying system.