Fast, Easy, and Accessible Evaluation of Memory Performance in Multi-node Computers

Daniel Robson

Most current systems use one pool of memory for the entire computer. When more CPUs are added, there is more contention for memory access across the single link to memory, slowing any computations as they wait for memory.

This style of system does not scale to scenarios with many CPUs, which is becoming increasingly common in many areas of computing.

**NUMA Systems**

- Core0
- Core1
- N0
- N1
- N2
- N3
- N4
- N5
- N6
- N7

- Single Bus Systems

**Why Develop a Software NUMA Model?**

There are currently no fast means of accurately modeling detailed interconnect and memory performance on these NUMA systems. Developers must now use detailed simulation, which can be the most accurate methods, but it is typically very slow and often cumbersome to setup and use.

There are no easily accessible means of rotating events or performance on a NUMA system to a developer analyzing applications. Aggregate event counts are very efficient, but are difficult to attribute precisely to regions of an application, lack event detail needed to perform detailed memory analysis, and are restricted to specific events.

**An Accessible and Flexible Technology Base**

Building on the Valgrind framework, we are constructing an efficient, accessible tool modeling the HyperTransport transport systems for developers called NUMAgrind. By dynamically adding measurement code into applications as they run, we reduce burden on end-users and allow very flexible analysis.

**Tool Feedback for Software Developers**

We allow developers access to more detailed forms of analysis than provided by hardware, while retaining sufficient information to tie events to specific code (source annotation).

Traditionally, simulators have traded speed for model precision, especially where event timing accuracy is concerned. While useful for system designers, software developers do not require as precise feedback, preferring more light weight tools.

Our tool will give data layout and source annotation feedback, allowing detailed evaluation of key data structures while retaining the ability to closely evaluate source code performance.

**Performance of Analysis Tools**

Our modifications to Valgrind which allow multithreaded analysis demonstrate substantial performance increase over the existing infrastructure we leverage.

Parallel analysis will prove key in retaining accuracy and efficiency of analysis for end users. This is necessary where ease of use and multiple analysis scenarios need to be investigated, decreasing turnaround time.

NUMAgrind demonstrates two orders of magnitude greater efficiency than full system simulation, while preserving the necessary model precision for our users.

**What Can We Do Currently?**

We can simulate very fine grained memory usage on arbitrary system topologies with an easily accessible tool. We aim to refine our analysis which will allow:

- Interconnect communication contention
- Memory placement efficiency
- Detailed aggregate event counts

All events can be traced back to causes in the application source code. This allows rapid identification of performance issues.