Parallel Computing Project(Design Document)

Submitted by

Shikhar Brajesh (2012A7PS114P)

Ravi Shankar Pandey (2012C6PS676P)

Scope

NUMA though a very popular architecture suffers from latencies due to intra-node asymmetric memory access costs and inter-node communication costs. Using a hierarchical load balancing approach, we can improve the performance of NUMA systems. For this we need to efficiently schedule tasks keeping in mind the topology of the cores such that communication between nodes is minimum. The paper presents an algorithm NUCO LB which takes the asymmetric memory access costs present on NUMA multi-core compute nodes, the interconnection network overheads, and the application communication patterns into account in its balancing decisions.

The algorithm is implemented using CHARM++ framework on three different type of NUMA architectures. The results for a set of sample programs are then discussed.

Implemented Design

NUCO LB is a greedy algorithm which assigns tasks kept in a priority order to the least loaded processor. The algorithm exploits information about the application data and the machine topology.

To implement our design we have performed scheduling on 4 NUMA nodes.

Each system has 4 cores. Processor used is i5-4570

Cache memory sizes- L1: 256 kB, L2: 1024 kB, L3: 6144kB

We are using AMPI along with Charm++ to provide a virtual view of the processors with different memory latencies.

(NOTE: AMPI is a wrapper library of MPI to provide support for MPI on charm++)

For each iteration of load balancing on the hybrid cluster following steps are done:-

- Sort the Processes(objects) by their predicted weight
- Compute the cost(c,t) to be used in the NucoLB for each processor in the cluster(cost(c,t) is cost of moving task ‘t’ to processor ‘c’).
- If the processor with minimum cost(c,t) is not the same as that of the current processor of task ‘t’ then migrate the task to processor c.
- Repeat until all the processes are over.

CHARM++ provides us with the following information about the nodes:

- Predicted load of each chare (process).
- Provides a mechanism to maintain and monitor processor load.
- Provides the details (such as amount of data to be sent) of the message transfer in case a process transfer is needed.
- Provides the details (such as core frequency) of the processors being used.

Although CHARM++ provides the necessary information about the parallel application, it does not provide any information about the intra-node topology. For that we have used HWLOC library. Since HWLOC does not provide information about the way NUMA nodes are connected together and how compute nodes are interconnected, we extend our approach to include NUCO Factor computation. To do so we need to compute and store Network Factor (done using OpenMPI's ping pong benchmark) and Numa Factor (calculated using LMbench benchmark).

Information Required:

Description of NUMA Machines: The algorithm discussed in the paper requires the details of the internal of the processing units such as:-

- communication cost of putting a particular task on the selected core.
- each core private caches details
- shared cache(if used)

As an example following are the details of the systems of IPC lab 6019 system:-

- each core has a L1 cache of 256 KB and L2 cache of 1024 KB
- all the core have a shared L3 cache of 6144 KB
- each system has a quad core(intel i5) CPU with a frequency of 3.20 GHz.

Benchmarking:

1. lmbench library of Ubuntu used.
2. pingpong test- To perform the ping pong test, we calculated the average time of 20 pings between each set of nodes.