A Hierarchical Approach for Load Balancing on Parallel Multi-core Systems

By
Raghav Jajodia 2012A7PS064P
Abhishek Nagpal 2012C6PS639P
Saurabh Satnalika 2012A7PS777P

NUCOLB, a topology-aware load balancer focuses on redistributing work while reducing communication costs among and within compute nodes. NUCOLB 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. Traditional Load balancers do not consider inter-node latencies for balancing the load between its nodes.

Design:
Given a NUMA topology of nodes, the problem is to design an efficient Load Balancer(NucoLB). NUCOLB will be executed on a single node in a cluster periodically, which will reassign the tasks to different cores, hence balancing the load.

At a given time, consider T tasks, C cores and a mapping of tasks to cores.
NUCOLB Algorithm:
Input: T set of tasks, C set of cores, M mapping of tasks to cores
Output: M’ mapping of tasks to cores

Algorithm
```
$M' \leftarrow M$
while $T \neq \emptyset$ do
$t \leftarrow k \mid k \in \arg\max_{l \in T} taskLoad(l)$
$T \leftarrow T \setminus \{t\}$
$c \leftarrow p, p \in C \wedge \{(t, p)\} \in M$
$coreLoad(c) \leftarrow coreLoad(c) - taskLoad(t)$
$M' \leftarrow M' \setminus \{(t, c)\}$
$c' \leftarrow p \mid p \in \arg\min_{q \in C} cost(q, t)$
$coreLoad(c') \leftarrow coreLoad(c') + taskLoad(t)$
```

$M' \leftarrow M' \cup \{(t, c')\}$

cost(c, t) - Cost to map task t to core c.
taskLoad(t)- Execution time (load) of task t.
coreLoad(c)- Sum of the loads of tasks mapped to core c.
lcomm(t, c)- Number of local messages received by task t.
rcomm(t, c)- Number of remote messages received by task t.
comm(t)- NUMA nodes of remote tasks communicating with task t.
node(c)- NUMA node of core c.
NUCO FACTOR(i, j)- Latency factor between NUMA nodes i and j

where
$\text{cost}(c, t) = \text{coreLoad}(c) + \alpha \times [\text{rcomm}(t, c) \times \text{NUCO FACTOR}(\text{comm}(t), \text{node}(c)) - \text{lcomm}(t, c)]$
NUCO FACTOR (i, j) = (Read latency from i to j)/(Read latency on i)

Time Complexity - O(nm) for real time scenarios and O(n(m^2)) in worst case
where n= tasks and m=cores

Implementation
We have implemented the load balancer using MPI.
Since it is difficult to get the information related to all the cores of a node, example, Core Load, Tasks assigned to a core etc, we assume the timewise information to be present in text-files. Executing populateDatabase.cpp, generates a random database.This database contains information about the Tasks (Task id ,taskLoad and Core assigned), Cores (Total Core Load and group of tasks assigned to the particular load). Also, the code generates this database MAX_ITERATION different times.

We have written the above mentioned algorithm in C++ using MPI. NUCOLB Algorithm runs periodically in cluster in all the nodes, and based on the new mapping of tasks to cores, nodes exchange tasks in cluster using MPI.

Before balancing the load, update() method updates the mapping data structure. At that particular point of time, it repopulates the core and task information from the database.

Here the Network factor is calculated using the averaged ping time among the nodes.