Efficient power utilization of a cluster using scheduler queues

Kalyana Chadalvada, Shivaraj Nidoni, Toby Sebastian
HPCC, Global Solutions Engineering
Bangalore Development Centre, DELL Inc.
{kalyana_chadalavada;shivaraj_nidoni;toby_sebastian}@dell.com

Abstract:

Clustering has become very common in high performance computing. The efficient utilization of all nodes in a cluster is managed using schedulers. A scheduler, however, will not be able to efficiently plan the optimal power usage of the cluster. Thus an efficient scheduling of jobs does not guarantee efficient utilization of power heat generation. We propose a mechanism by which the scheduler can plan efficient job allocation along with optimal power utilization.

Motivation:

In the modern day data center, performance alone is no longer the only yard stick on which a system is designed. Power consumption and HVAC requirements have gained a lot of significance. What system designers are looking at as the new metric is “performance per watt” rather than “performance per price” alone. Several approaches exist to reduce the power consumption of systems. But these approaches alone are not sufficient in a cluster environment. Current power saving mechanisms are not sufficient by themselves to achieve appreciable power savings in a cluster environment. For example, nodes in the cluster cannot be configured with various time based sleep states. If a system is configured to suspend to disk after being idle for 45 mins will show up as a dead node as soon as the system does that. Current cluster management software stacks and schedulers do not understand that the system is in a power saving state.

The job scheduler in a cluster is aware of the current state of the queue at any given point of time. Depending on the queue SLA’s, it can predict to some extent, the system usage. By looking at the cluster usage statistics, and combined with a learning algorithm, scheduler can compute which nodes will be used in the near future. The nodes which are not seen as required can be marked and powered off by scheduler. When a requirement arises, scheduler can power up the inactive nodes and utilize them for job scheduling. This will improve power consumption and reduce heat generated by a cluster, which is an area of concern in a modern data center.

Description of proposal:

New generation of systems now implement one or more types of out of band access mechanisms based on IPMI/BMC standard. This gives the end user ability to remotely manager systems without paying a premium for this capability. It also allows programmatic control of systems based on events or triggers. Modern processors, systems and operating environments also are better at power management. Some examples include AMD PowerNow!, Intel SpeedStep, demand based switching, bus / frequency throttling, ACPI compliance and the like. The processors themselves are able to support multiple sleep states to reduce power consumption. For example, if a UNIX bases system is currently active at init level 5, it has a host of services active that are needed when the system is being actually used. So, the amount of memory actively used and the number of components actively used is more than at init level 2. So, we need a complimentary mechanism that can work along with existing technologies and help achieve better power management without too much overhead.

The proposal is to provide extend the cluster job scheduler to achieve the following:

- Implementing a learning mechanism to understand cluster usage patterns
- Aware of an out of band connectivity to the cluster nodes like IPMI/BMC or any other ILO mechanisms
- Able to introduce new "node status" values

Traditionally, the node status is broadly marked as follows:

- Available – available for use
- Closed – not available for use
- Dead – unable to reach the system

We plan to extend the above to include the following:

- Sleeping – system is put in a stand by mode (suspend to memory)
- Hibernating / switched off – system is switched off or put in hibernation (suspend to disk)

The scheduler is provided access to the cluster node's out of band network in addition to the administration fabric. For the sake of discussion, let us consider using IPMI/BMC as the OOB fabric. Figure 1 illustrates a typical cluster setup. Since this is a standards based implementation, it would be easier for the scheduler developers to provide support of the same. The IPMI specification provides "remote power on" and "remote power off" commands without the need for an OS agent.

At set intervals, the job scheduler will inspect the state of the queues and the nodes. Based on cluster usage patterns and/or administrator set exceptions, the scheduler will draw up a set of nodes that can be put in various sleep states. This can be achieved in several ways. Let us first describe probable power saving states:

- OS based
    - Move to a lower init level to shut off un-necessary services and reduce memory / processor / NIC/ hard disk usage
    - OS initiated sleep states – suspend to memory / suspend to hard disk
    - Turn off display / network / disk
    - Force processor to move to several native sleep states through API's where available – Ex:C0,C1,C2,C3
    - Force multiple cores / processors to go offline
- OOB fabric based
    - Graceful shutdown

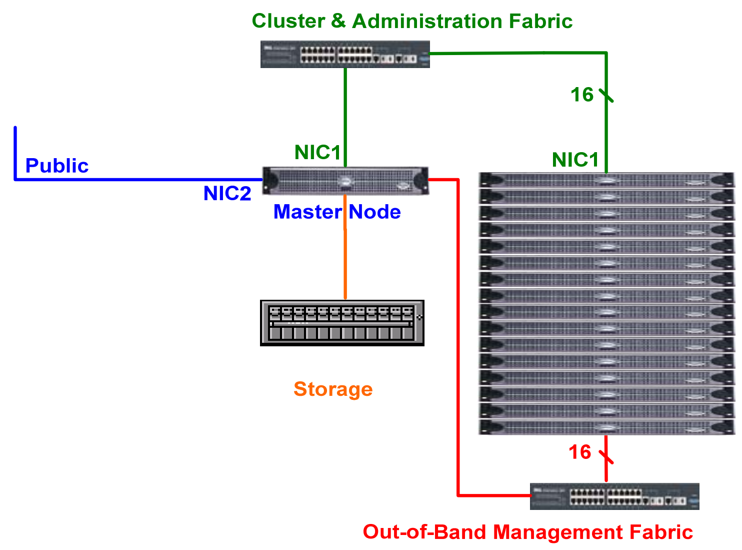
Cluster & Administration Fabric
16
Public
NIC1
NIC1
NIC2
Master Node
Storage
16
Out-of-Band Management Fabric

Figure 1. Typical cluster setup and components

The scheduler daemon on the nodes can fire off appropriate system calls and/or API calls to achieve all OS based power saving mechanisms. In case of scheduler daemon not reachable due to the system sleep state; scheduler can utilize the OOB fabric to bring the system back up. In case of a multi core / multi processor system, the easiest method is to force all cores and processors to go offline and have one core active. Vendor APIs and OS support needs to be available for this.

The scheduler will have a threshold value for each state. The system can be moved to a lower sleep state only if it has spent a set value of time in the current state. For example, a fully functional system can be moved to lower init level only when it has been idle for 30 mins and there is a computed forecast of at least 15 mins idle time. Similarly a system can be put in suspend to disk or shutdown only if it has spent atleast 60 mins in suspend to memory and there is a computed forecast of at least 15mins time. This will avoid node thrashing, a state in which the system is moving across various sleep states in small time intervals and not providing a significant power saving. Administrators can also set exceptions on certain systems, queues to guarantee a minimum SLA / high priority nodes.

A simple flow chart of the scheduler decision process is illustrated in figure 2.

When the scheduler detects that a system is needed, it will look up the state of the node in its table and if the system is in a sleep state; the scheduler will put the job on hold till it brings the system back online. As soon as the system is brought to this active state, the job is scheduled on the node.

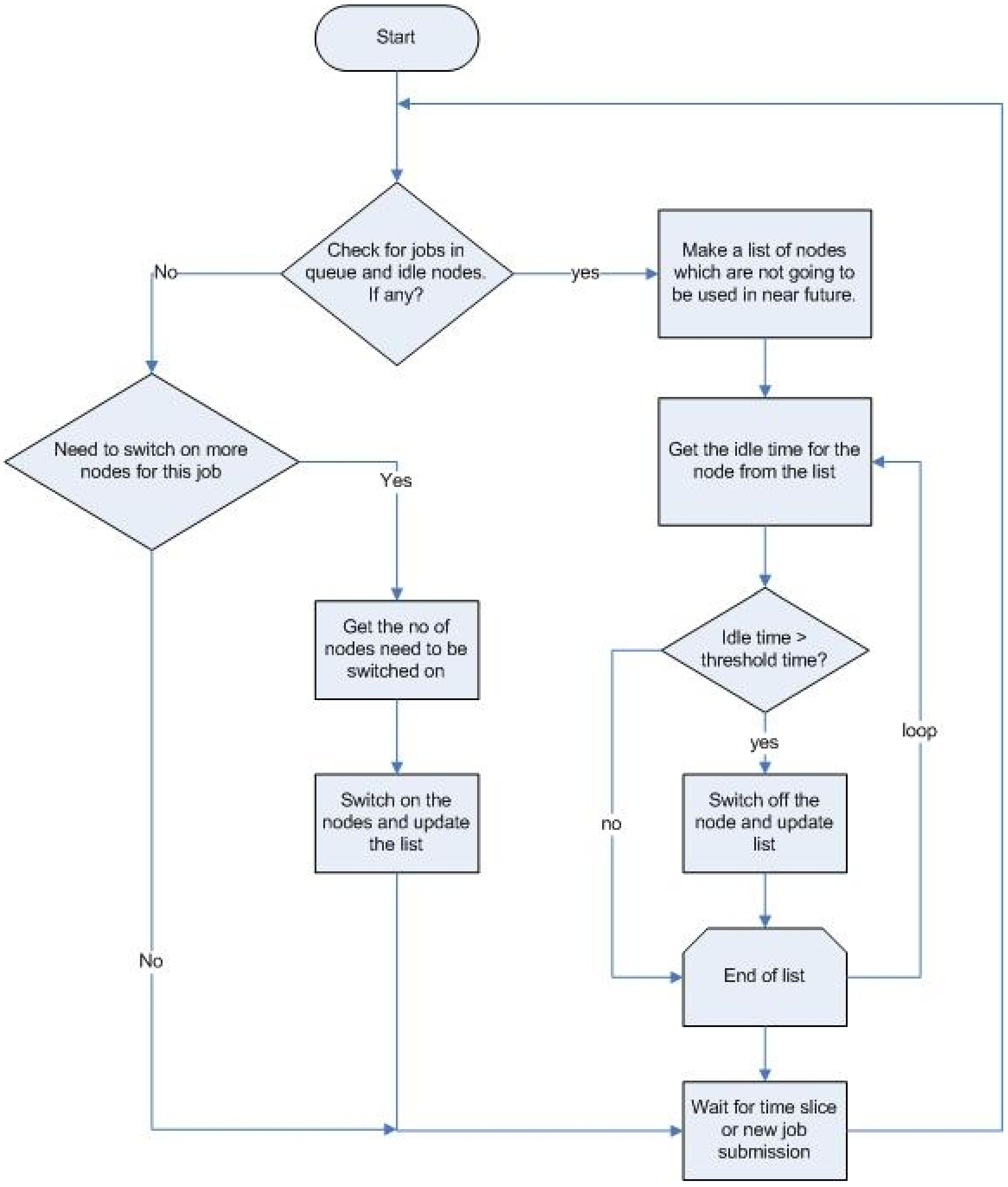
Start
Check for jobs in queue and idle nodes. If any?
yes
No
Make a list of nodes which are not going to be used in near future.
Need to switch on more nodes for this job
Yes
Get the idle time for the node from the list
Get the no of nodes need to be switched on
Idle time > threshold time?
yes
no
loop
Switch on the nodes and update the list
Switch off the node and update list
No
End of list
Wait for time slice or new job submission

Figure 2. Slow chart of scheduler decision process

Future Work:
Implement a minimal proof of concept on existing open source schedulers like OpenPBS. A sample implementation will help us measure the probable power savings. We plan to support IPMI based OOB and LINUX supported power saving states. Possible changes also include modifying the cluster management / monitoring software like ROCKS / Ganglia to understand power saving states.

Conclusion:

We are proposing a minor modification to the current scheduler which will improve power utilization in a cluster. We would like to make the following enhancements moving forward:

- Use learning algorithms to get understand the nodes utilization patterns in a cluster and make the decisions on the basis of utilization statistics.