Unobtrusive Power Proportionality for HPC Frameworks *

Arka Bhattacharya
University of California, Berkeley
arka@eecs.berkeley.edu

David Culler
University of California, Berkeley
culler@cs.berkeley.edu

ABSTRACT

Building power proportional High Performance Computing (HPC) clusters comprising of servers which are not power-proportional is a well-studied problem, and has the potential to provide large energy savings [2]. However, a large emphasis on maintaining cluster uptime disincentivizes system administrators from deploying prior research techniques that introduce changes to existing software configurations, modify the existing cluster job management framework, change user job submission procedures, or fail in unpredictable ways due to frequent server power cycling [3].

We present *Hypnos*, a meta-system that tackles the challenge of implementing power proportionality **unobtrusively** in an HPC cluster with an existing job management framework. Hypnos makes no changes to the existing cluster software or network stack, and uses only the standard interfaces exposed by the existing cluster framework to (a) obtain server state and job information, (b) add/remove servers from the existing framework's purview, (c) infer the cluster's scheduling logic, and (d) handle reliability challenges when servers fail to run jobs, boot up, or race conditions develop between Hypnos and the existing cluster scheduler.

We evaluated Hypnos by deploying it on a production HPC cluster running the framework - Torque [4]. Hypnos was able to achieve a 36% reduction in energy consumption (compared to an optimal of 37.5%) while circumventing over 1500 network and software faults over a 21-day deployment.

Categories and Subject Descriptors

C.4 [**Performance Of Systems**]: Reliability, availability, and serviceability; C.5.5 [**Performance Of Systems**]: Computer System Implementation—*Servers*

*For a full version of this work, refer to [1]. This work was funded by NSF Grants CPS-0932209 and CPS- 0931843.

e-Energy'14, June 11–13, 2014, Cambridge, UK.
ACM 978-1-4503-2819-7/14/06.
http://dx.doi.org/10.1145/2602044.2602085 .

Keywords

Power Proportionality; Unobtrusive; Meta-system; High-Performance Computing; Reliability

1. HYPNOS DESIGN

Hypnos uses the observation that most existing HPC job management frameworks (e.g Torque, Oracle Grid Engine, IBM Load Sharing Facility) expose three interfaces which allow unobtrusive power management - (a) An interface to add / remove servers from the cluster framework's purview (b) An interface exposing server state information, and (c) An interface exposing details of jobs submitted to the cluster and their constraints. Hypnos resides on the existing cluster framework's master node, and comprises of three modules - the Framework Interface Layer, a Server State Machine(SSM) and Failure Handler(FH) for each server, and the Power Management Algorithm(PMA).

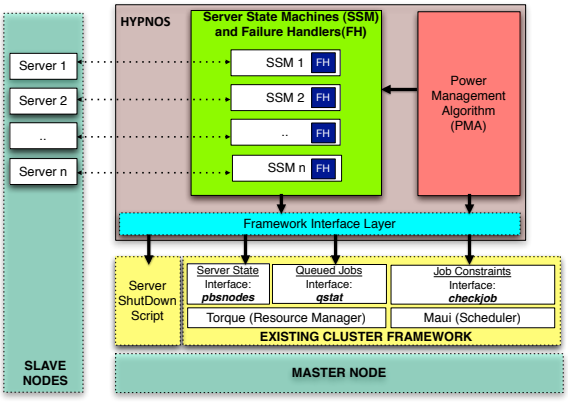

Figure 1: Hypnos System Design. The interfaces Hypnos uses on an HPC cluster running Torque is *pbsnodes*, *qstat* and *checkjob*. Other HPC frameworks have analogous interfaces which Hypnos could use.

Framework Interface Layer: This layer obtains the information required by the other Hypnos modules from the interfaces exposed by the cluster's job management framework, and can be re-written for different cluster frameworks. The Torque-specific interfaces used in our implementation is shown in Figure 1.

Power Management Algorithm(PMA): The PMA implements a wakeup and shutdown control loop using the

job information obtained through the Framework Interface Layer and the server state information reported by the Server State Machines. The PMA wakes up servers if existing queued jobs cannot be bin-packed on to the set of already powered-up servers (or the set of servers that are currently waking up). The shutdown control loop shuts down servers in case they have been idling for more than a user-specified threshold, provided powering them down does not affect the cluster's minimum spinning-reserve[1] capacity.

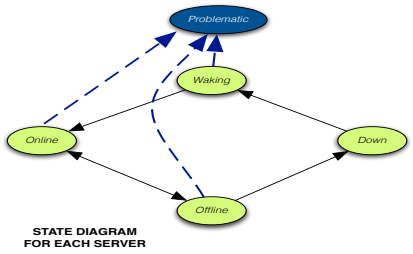

Figure 2: State Machine maintained for each server

Server State Machine(SSM), Failure Handler(FH): Hypnos utilizes the information obtained through the cluster framework's interfaces and the state transitions ordered by the PMA, to implement a Server State Machine (SSM) and a failure handler (FH) for each server. Each server can be in 5 possible states(Figure 2):

(a) `Online:` signifying that the server is powered up and is either executing jobs or is idle;

(b) `Down:` signifying that the server is powered off;

(c) `Offline:` a state a server goes through before it transitions to the `Down` state from the `Online` state. If a job has been scheduled on it due to a race condition between Hypnos and the existing cluster framework, the server is brought back to the `Online` state.

(d) `Waking:` is an intermediate transition state between `Down` and `Online` to account for the time elapsed between servers being commanded to power up and when they become ready to execute jobs.

(e) `Problematic:` signifying that Hypnos has inferred either (a) a failure which renders the server incapable of executing jobs, or (b) some discrepancy between the server's state as maintained by Hypnos and the information obtained from the cluster framework's interfaces. Such inference may happen when a server was presumed by Hypnos to be in the `Online`, `Offline` or `Waking` states. Depending on the state a server transitions to `Problematic` from, Hypnos considers the possible reasons for such a discrepancy, and gracefully handles them through that server's Failure Handler (FH) module. (For details, see [1])

Hypnos thus achieves *unobtrusiveness* by virtue of its meta-system design, where it sits on top the cluster's master node and only uses the interfaces exposed by the existing cluster framework. Hypnos achieves *reliability* by maintaining a state-machine for each server, and periodically corroborating its presumed state of the server with the information obtained from the cluster framework's interfaces. Hypnos is also *extensible*, due to the decoupling of the Framework Interface Layer, the Power Management Algorithm, and the server-specific SSM and FH modules. The wakeup and shutdown control loops in the PMA can be optimized in isolation (without having to worry about reliability) to take into account cluster-specific workload features such as its diurnal patterns or its burstiness (e.g [5]).

[1]The spinning reserve is a set of idle servers kept powered up to service smaller/interactive HPC jobs

2. EVALUATION

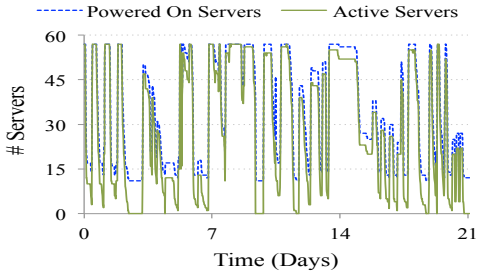

Figure 3: Number of servers kept powered-on closely matched the number of active servers when Hypnos was deployed

Hypnos was evaluated over 21 days on a 57-server cluster consisting of 51 Dell PowerEdge 1850 servers (192W idle / 292W peak), and 6 Dell PowerEdge 1950 servers (253W idle / 387W peak). It achieved a 36% energy savings (37.5% ideal), while serving over 3650 jobs and subverting over 1500 failures, such as a server failing to load essential networked services (e.g the Networked File System) or local filesystem errors which caused the inability of a powered-up server to run jobs. Figure 3 shows that Hypnos was able to power down idle servers, thus, closely matching the number of powered-up servers to the number of *active* servers (servers running jobs).

3. REFERENCES

[1] Arka Bhattacharya and David E. Culler. Hypnos: Unobtrusive power proportionality for hpc frameworks. Technical Report UCB/EECS-2014-29, EECS Department, University of California, Berkeley, Apr 2014.

[2] Luiz André Barroso and Urs Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, December 2007.

[3] Edmund B. Nightingale, John R. Douceur, and Vince Orgovan. Cycles, cells and platters: an empirical analysisof hardware failures on a million consumer pcs. In *Proceedings of the sixth conference on Computer systems*, EuroSys '11, pages 343–356, New York, NY, USA, 2011. ACM.

[4] Torque Resource Manager. `http://www.adaptivecomputing.com/products/open-source/torque/`.

[5] Kai Wang, Minghong Lin, Florin Ciucu, Adam Wierman, and Chuang Lin. Characterizing the impact of the workload on the value of dynamic resizing in data centers. In *ACM SIGMETRICS Performance Evaluation Review*, volume 40, pages 405–406. ACM, 2012.