Dynamic Power Management for Hardware Over-provisioned Systems

Daniel Ellsworth
dellswor@cs.uoregon.edu

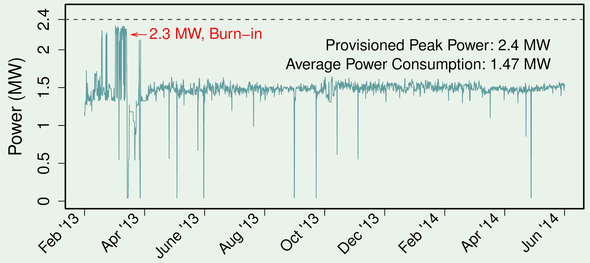
Problem Statement

Given technology allowing per component power caps to be set, how can the power resource be safely managed across a shared hardware over-provisioned HPC cluster?

Background

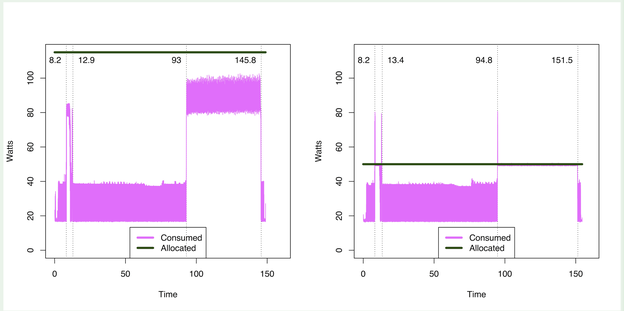
Data center design and power budgets target the maximum draw to protect the hardware from damage. The aggregate power used by a cluster can be significantly less than the maximum theoretical power draw. A hardware over-provisioned system would include extra nodes to use the surplus power for additional computation.

Total Power Consumption of the BG/Q Vulcan Supercomputer Feb 2013 to Jun 2014



T. Patki, et. al. Practical Resource Management In Power-constrained, High Performance Computing. HPDC 2015

Trying to schedule power at job start time, like nodes are currently scheduled in most clusters, will likely result in inefficient power utilization. Many applications have phased consumption behavior and, to avoid performance degredation, a cap set at job start would need to be set to the maximum consumption experienced at any time during the job.



Power Scheduler Invariant

At an absolute minimum, a power scheduler for a hardware over-provisioned system must guarantee that the total power consumption across all components is less than the system wide power limit. If per component power allocations can be set, it is sufficient for the sum of the allocations to be less than the system wide power limit.

$$\forall(t,i), c_i^t \leq a_i^t \text{ and } \forall t, \sum_{i=1}^{n} a_i^t \leq L \implies \forall t, \sum_{i=1}^{n} c_i^t \leq \sum_{i=1}^{n} a_i^t \leq L$$

Static Power Schedule

$$\forall(t,i), a_i^t = \frac{L}{n}$$

Job Power Schedule

Let a_j be the estimated power a job will consume.

Schedule concurrent jobs only when $\forall t, \sum_{j=1}^{J} a_j \leq L$

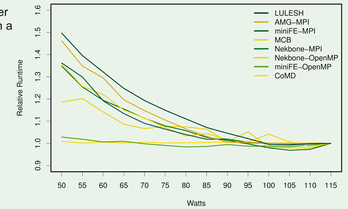
At job launch, set a_i for all nodes in job j such that $\sum a_i = a_j$

Dynamic Power Schedule

At any time t set new a_i such that $\sum_{i=1}^{n} a_i^t \leq L$

Simulation Model

Runtime generally increases polynomially under progressively lower power caps. The effect of power capping only occurs when the cap is set lower than a computation's uncapped consumption.

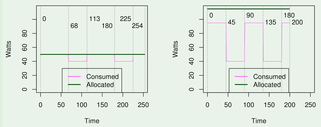


$$I_s = \int_s^e \sqrt{\frac{b - S_-}{S_+ - S_-}}\,\mathrm{d}t \quad , \quad E_s = \int_s^e b\,\mathrm{d}t$$

$$I_p = \int_s^e \sqrt{\frac{w_{(t)} - S_-}{S_+ - S_-}}\,\mathrm{d}t \quad , \quad E_p = \int_s^e w_{(t)}\,\mathrm{d}t$$

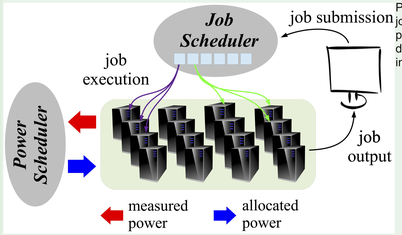
I_s	Socket instruction work possible
I_p	Program instruction work
E_s	Socket energy allocated
E_p	Program energy consumed
$w_{(t)}$	Program power consumption at time t
s	Interval start time
e	Interval end time
b	Socket power bound
S_-	"Idle" socket power consumption
S_+	Maximum socket power consumption

The simulation models a processor as being able to complete an amount of instruction work over an interval based on the power cap. A program is modelled as a function of watts over time on an uncapped processor. A program finishes execution when all instruction work is complete.



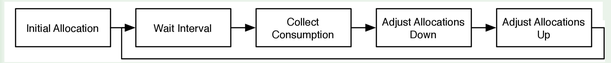
Simulation behavior for computations with bound and unbound intervals due to processor capping are like experimental results.

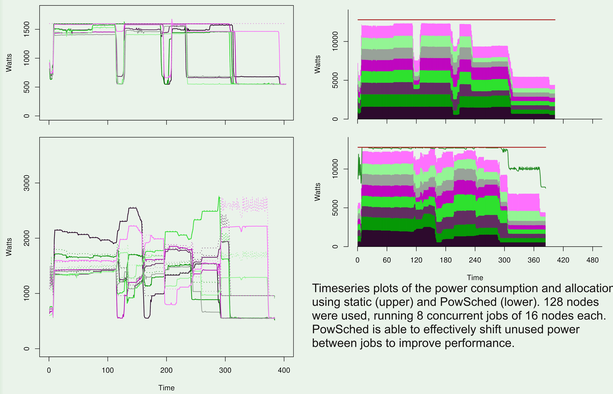
Power Scheduler - PowSched



Power scheduling must take priority over jobs since exceeding the system-wide power bound could result in physical damage. PowSched operates completely independently of the job scheduler.

PowSched is unaware of jobs or other partitioning of the system. At each interation, consumption is collected from all components and new allocations are applied to all components.





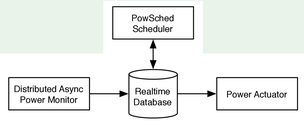
Timeseries plots of the power consumption and allocation using static (upper) and PowSched (lower). 128 nodes were used, running 8 concurrent jobs of 16 nodes each. PowSched is able to effectively shift unused power between jobs to improve performance.

Experiment	Runtime	Stddev	Improvement	kj Alloc	Stddev	kj Used	Stddev
115W static	278.26	9.57		8191.85	281.75	4007.80	97.99
115W dynamic	276.24	4.84	0.7%	5474.75	52.56	3977.02	36.74
90W static	284.63	3.20		6571.76	72.75	3984.68	30.32
90W dynamic	277.13	5.04	2.6%	5339.11	66.21	3979.78	47.356
70W static	323.83	4.90		5829.02	86.82	3904.29	34.08
70W dynamic	278.02	4.97	14.1%	4638.32	68.91	3984.80	37.77
50W static	401.76	5.47		5178.29	72.59	3937.65	37.52
50W dynamic	371.92	13.23	8.7%	4562.48	124.44	4015.64	79.36

The above table shows the averaged results of ten 128 node runs of the same workload. The workload was composed of eight concurrently executing 16 node benchmarks. Static refers to machine static power allocation. Dynamic refers to PowSched's dynamic allocation.

Current Work

Is tight coupling required between collection, decision, and actuation? Preliminary results suggest no however performance is decreased.



All three major tasks are asynchronous and independent. The database provides the shared view of system state for components updating state or making decisions.

Experiment	Runtime	Improvement	kj Alloc	kj Used
115W static	278.26		8191.85	4007.80
115W dynamic	281.18	-1.0%	5662.175	3930.92
90W static	284.63		6571.76	3984.68
90W dynamic	284.49	0.0%	5547.65	3961.87
70W static	323.83		5829.02	3904.29
70W dynamic	288.58	10.8%	4830.91	3972.80
50W static	401.76		5178.29	3937.65
50W dynamic	381.33	5.1%	4736.71	3822.73

This table compares static and decoupled runtimes for the 128 node experiment referenced in the PowSched column.

Experiment	Runtime	Improvement	kj Alloc	kj Used
115W static	635.28		18735.93	8423.71
115W dynamic	651.29	-2.5%	12168.17	8529.40
90W static	631.92		14572.81	3984.68
90W dynamic	645.16	-2.1%	11947.70	8515.70
70W static	686.06		12312.24	8217.15
70W dynamic	655.40	4.5%	10825.60	8390.14
50W static	832.01		10656.16	7525.93
50W dynamic	737.56	11.4%	9250.22	7573.55

Static and decoupled runtimes for a 128 node experiment using a different a workload.

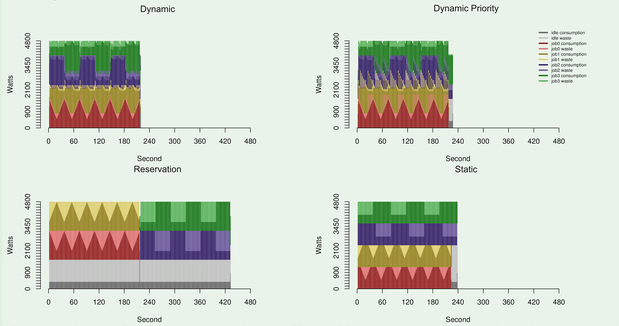
Experiment	Runtime	Improvement	kj Alloc	kj Used
115W static	638.21		38025.55	16742.65
115W dynamic	650.97	-2.5%	24311.96	17001.13
90W static	631.31		29175.82	16266.65
90W dynamic	654.49	-2.1%	24048.24	17101.51
70W static	681.91		24828.42	15168.44
70W dynamic	654.24	4.5%	21433.79	16525.66
50W static	849.91		21801.45	14956.87
50W dynamic	759.09	11.4%	18854.32	15095.90

Static and decoupled runtimes for a 256 node experiment.

What are the behaviors of different strategies?

Dynamic - Power is dynamically set per socket during runtime without knowledge of the job running on the socket.
Reservation - Power per socket is set to the estimated max consumption at job launch. Jobs are only launched if sufficient power is available.

Dynamic Priority - Power is dynamically set per socket during runtime. Sockets for the highest priority job always get increased first.
Static - The system power cap is divided evenly across all sockets and power estimates are not considered at job schedule time.



All simulated jobs use 8 nodes and consume between 30 and 100 watts per socket. The simulated cluster has a total of 32 nodes and a system power cap of 4800 watts. Sockets support a minimum cap of 50 watts and maximum of 115 watts.

Acknowledgements

This work would not have been possible without the mentorship of Allen Malony, Martin Schulz, Barry Rountree, and Tapasya Patki. Part of this work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Work at the University of Oregon was supported by the DOE Office of Science, through a Sub-Contract No. 3F-32643 from the University of Chicago, Argonne, LLC (as operator of Argonne National Laboratory), under Prime Contract No. DE-AC02-06CH11357.
LLNL-POST-704637