Using Batch Controlled Resources to Support Urgent Computing

Rich Wolski, Dan Nurmi, John Brevik, Graziano Obertelli

Computer Science Department

University of California, Santa Barbara

UCSB

Batch Controlled Machines

- **Batch system**
  - **Batch scheduler**
    - **Scheduler (Maui, Catalina, SGE)**
    - **Provisioner (Torque, PBS, LSF, SGE)**
  - **Job script**
    - **describes the program, number of nodes, the maximum occupancy duration, etc.**
  - **Site administrator**
    - **Sets priorities by determining scheduling policy**
    - **Partially advertises policy in the form of separate queues for the users**
  - **Allocation**
    - **Each job's machine occupancy is charged against an account**
    - **Aspect ratio is not considered (e.g. units are node-hours)**
    - **Scheduling policy can set charging rates**
- **Three approaches to providing support for urgent computing**
  - **Over provisioning (obvious solution)**
  - **Pre-emption (logical solution)**
  - **Scheduling ("out there" solution)**

Over provisioning

- **Keep hot spares**
  - **The "emergency provisions" approach**
- **Expensive**
  - **Maintenance, system administration, etc.**
  - **Hard to track technology improvements**
- **Software is not stationary**
  - **If the system gets upgraded, new upgrades need to be tested, patched, etc.**
  - **If the system remains static, new applications developed in maintained environments won't run**
- **One possibility: use virtualization to allow the old (but tested) environment to run in a development setting**
  - **Makes it to ensure that the software will work on the hot spare when needed**

Pre-emption

- **Most straight-forward approach is to kill running jobs when urgent jobs arrive**
  - **Often the jobs being pre-empted are long-running => killing them can result in a substantial loss of productivity**
    - **Kill policy: kill jobs from youngest to oldest until enough nodes are available**
    - **May not be enough young jobs to free up the needed nodes**
      - **Most jobs in the systems are short and narrow**
  - **Pre-emption cluster (due to David McWilliams at NCSA)**
    - **Put in a cluster where users know their jobs can be pre-empted and charge less to use it for the additional risk**
    - **Set scheduling policy to encourage short, narrow jobs**
      - **Work loss is less**
      - **Takes "small" work off the other machines**
      - **Not sure what utilization looks like**

Checkpointing and Pre-emption

- **Transparent checkpointing**
  - —**Requires OS support**
  - —**Space intensive since entire memory state may need to be saved**
  - —**Checkpoints not portable across architectures**
- **Semi-transparent (ala Condor)**
  - —**Requires compiler and library support**
  - —**Full functionality is hard (threads, sockets, forking, etc.)**
  - —**Checkpoints not portable across architectures**
- **Application-level**
  - —**Usually requires programmer to put in explicit calls**
  - —**Automatic systems often impose a performance penalty**
  - —**Portable across architectures**
  - —**Most appropriate for "packages" (e.g. NAMD)**
- **Open question: what should a pre-empted job be charged?**

Pre-emption Experiment

- **Create a 128 node pre-emption cluster inside the NCSA DTF**
- **Provide support for Condor glide-in**
- **Two classes of Condor jobs**
  - **Pre-emption only**
    - **Will only be scheduled in the pre-emption cluster**
    - **Next-to-run status when nodes free if pre-empted**
    - **Run for free**
  - **Migration-needed**
    - **Runs in the pre-emption cluster until pre-empted**
    - **Restarted on first-available**
    - **Charged a reduced rate**
- **Non-Condor jobs are killed**
- **Pre-emption priority is youngest to oldest**

Scheduling

- **SPRUCE -- Next-to-run scheduling service**
  - **—Predict the bounds on time to start a next-to-run job on various machines**
  - **—Use multiple next-to-run requests to meet a specific probability target**
- **VARQ -- Virtual Advanced Reservations**
  - **—Use QBETS batch queue prediction system to create a statistical reservation**
  - **—Reservations function as an overlay so local scheduler need not participate**
- **Hard reservations and conservative back filling**
  - **—It is possible to implement a kind of hard reservation if conservative backfilling is used**
  - **—May not be harmful to non-urgent workload**

Predicting Next-to-run Time

- **Next-to-run semantics are unambiguous**
  - —**Job specifies number of nodes (run time is indefinite)**
  - —**Machine "drains" until that many nodes are available**
  - —**Job is released and machine continues to operate**
- **Trace-based faster-than-real time simulation**
  - —**Provisioner job logs show node and processor occupancy**
  - —**At any time point in a trace, it is possible to answer the question**
    - ***From now, how long until X nodes are available?***
- **Simulator says that the times *were* -- not what they will be**
  - —**Need a predictive methodology that can forecast what the next-to-run time *will be* based on what it has been**
  - —**Simulator avoids the need to constantly run next-to-run jobs**

QBETS: A New Predictive Methodology

- **New quantile estimator invention based on Binomial distribution**
  - **Requires carefully engineered numerical system to deal with large-scale combinatorics**
- **New changepoint detector**
  - **Binomial method in a time series context is difficult**
  - **Need a system to determining**
    - **Stationary regions in the data**
    - **Minimum statistically meaningful history in each region**
- **New clustering methodology**
  - **More accurate estimates are possible if predictions are made from jobs with similar characteristics**
  - **Takes dynamic policy changes into account more effectively**

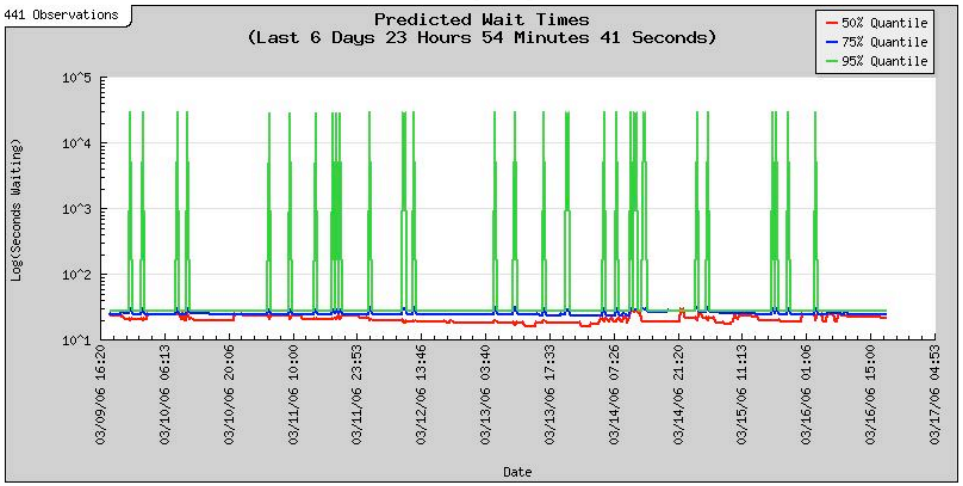
Real Time Prediction

- **Methodology:**
  - —**Use the Network Weather Service to gather queue data**
    - **Built for robust, accurate, real-time forecasting**
  - —**Stream the queue logging data into the simulator**
    - **Does not require that we actually run a SPRUCE job**
  - —**Forecast the bounds on delay using QBETS using a dedicated cluster at UCSB**
    - **QBETS is heavier weight than previous NWS forecasters**
    - **We have it heavily instrumented so that we can do constant QA**
  - —**Publish the results in real-time via a number of different interfaces**
    - **Only HTML is currently enabled to keep from confusing the Web Service clients**

See it In Action

- **http://nws.cs.ucsb.edu/batchq**

ucteragrid[dque] Nodes: 1-4 [more]



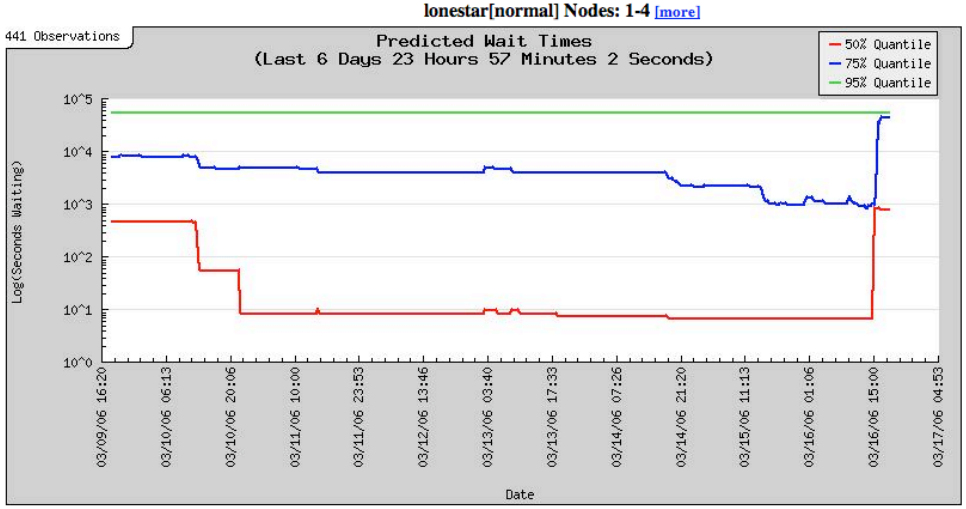
Quant. Prediction

50%	23.0 Seconds
75%	26.0 Seconds
95%	29.0 Seconds

Graph: jpeg png ps

Last Observation: Mar 16 2006, 18:17.04

lonestar[normal] Nodes: 1-4 [more]



Quant. Prediction

50%	821.0 Seconds
75%	46171.0 Seconds
95%	58249.0 Seconds

Graph: jpeg png ps

Last Observation: Mar 16 2006, 18:17.50

UCSB

Predicting Things Upside Down

- **Deadline scheduling: *My job needs to start in the next X seconds for the results to be meaningful*.**
  - **—Amitava Mujumdar, Tharaka Devaditha, Adam Birnbaum (SDSC)**
    - **Need to run a 4 minute image reconstruction that completes in the next 8 minutes**
- **Given a**
  - **—Machine**
  - **—Queue**
  - **—Processor count**
  - **—Run time**
  - **—Deadline**
- ***What is the probability that a next-to-run job will meet the deadline?***
- **http://nws.cs.ucsb.edu/batchq/invbqueue.php**

String and Chewing Gum

- **Building a reliable next-to-run predictor using QBETS and the NWS remains a challenge**
  - **Log quality is *terrible***
    - **Scheduler and provisioner logs are designed for accounting and not inference**
      - **Slop is okay as long as it is not a lot of slop**
      - **Crashes/freeze ups don't matter**
    - **There may be good reasons to fuzz precision**
      - **Having a log that requires interpretation allows for there to be an interpreter**
      - **Inferences are difficult for non-experts to understand**
    - **Some provisioners do not log the necessary information**
  - **Machine status information is not available in real-time**
    - **No site maintains a running node-status log**
    - **Not clear one can be built easily**
      - **We have experiment with statistical inference for this purpose**

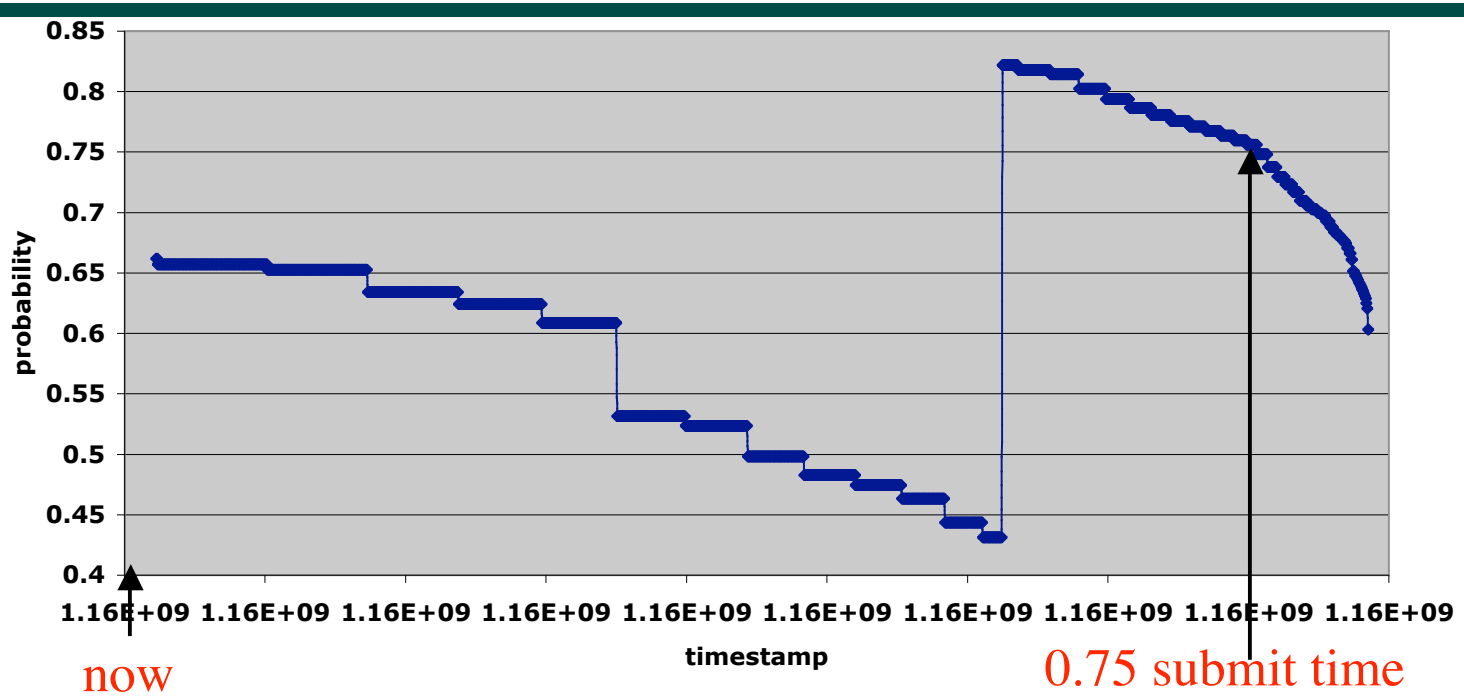
Virtual Advanced Queue Reservations

- **Using QBETS probability estimator it is possible to implement a reservation overlay: VARQ**
  - **—Sites run best effort schedulers without reservations**
  - **—A job specifies a**
    - **Time deadline (reservation start time)**
    - **Execution duration**
    - **A success probability**

      More certain => reservation is less likely
  - **—Submission agent queries QBETS to determine the best time *in the future* to submit the job so that**
    - **The job will be running at the deadline**
    - **The probability of doing so is at least as great at that specified**
  - **—Job may start early, but if it does, it can simply sleep until the deadline**
    - **Early time is lost allocation (cost)**
    - **We have a tool for automating the synchronization of the start**

UCSB

VARQ Probability Trajectory



- **75% is the target probability**
- **356 total requests**
- **257 total batch submissions**
  - **99 requests resulted in initial 'not possible' response**
- **192 slots successfully acquired**
- **257 * .75 = 193**

UCSB

FAQ

- **What happens if everyone uses these predictions? Will it be stable?**
  - **Maybe**
  - **We do not consider jobs in queue**
  - **Automatic schedulers may cause destabilization**
- **What about autocorrelation (you idiot)?**
  - **Difficult to compute in this space**
    - **Error-prone for non-stationary series**
    - **Queues reorder the series**
  - **Autocorrelation is and is not an issue**
    - **Quantile estimation and clustering algorithm are relatively robust to autocorrelation**
    - **Change-point detector computes uses the autocorrelation it computes on the fly**
- **Not a guarantee since it can fail**
  - **All guarantees come with a failure probability**

Musings

- **Urgent computing support on current systems is possible using a variety of extant software technologies and policies**
  - —**Checkpoint/restart**
  - —**Condor**
  - —**Accounting incentives**
- **It is possible to build inference tools that permit useful scheduling decisions**
  - —**Predicting next-to-run times is possible using on-line simulation**
    - **Non-intrusive solution**
  - — **scheduler/provisioner logging needs to be MUCH better for this to generalize**
- **Statistical virtualization could be a powerful tool for urgent computing**
  - —**Reservation support as an overlay => non-intrusive**
  - —**Success probability => joint probabilities are easy to compute**

Status

- **QBETS, VARQ, and SPRUCE**
  - —**QBETS in production use on TG and other systems**
  - —**VARQ is a robust functional prototype**
  - —**SPRUCE predictions are not quite ready for prime time**
    - **Problems with the logs are the obstacle**
- **What's next?**
  - —**Bandwidth => Can we do the same thing for data staging?**
- **Thanks**
  - —**VGrADS project (K. Kennedy)**
  - —**NSF SCI, NSF NGS, TiTech, SDSC, TG GIG, TACC, NCSA, Argonne**
- **rich@cs.ucsb.edu**