Local Scheduling for Volunteer Computing

David P. Anderson
U.C. Berkeley Space Sciences Lab

John McLeod VII
Sybase

March 30, 2007
BOINC projects and volunteers

- Projects
  - Einstein@home, CPDN, etc.
  - independent, separate servers
  - may be down or not have work
- Volunteers
  - run BOINC client on PC
  - “attach” client to projects
    > 1 project recommended
  - client queues jobs
- Scheduler RPCs
  - initiated by client
  - request X seconds of work
Local scheduling policies

- CPU scheduling
  - what job(s) to run
- Work fetch
  - when to get more jobs
  - what project to get them from
  - how much work to ask for
Inputs to the policies: host

- Host hardware
  - # CPUs, FP/int benchmarks
  - RAM size
- Host availability
  - % time BOINC is running and allowed to compute
  - % time network connected
**Inputs: user**

- User preferences
  - project resource shares
  - whether to compute when user active
  - time-of-day limits
  - CPU throttling
  - RAM usage limits while active/idle
  - Network connection interval
  - SchedulingInterval (~1 hour)

- User controls
  - suspend/resume all computation
  - suspend/resume/attach/detach project
  - suspend/resume/abort job
Inputs: static job parameters

- Each job has:
  - deadline
  - estimate of FP ops
  - limits on disk/RAM/FP ops
- Redundant computing and deadlines:
  - if a job is not reported by its deadline, it’s increasingly likely that it will get no credit, and have no value to the project.
**Inputs: dynamic job parameters**

- Checkpointing
  - requested periodically by BOINC client
  - reported by application
- Fraction done
  - reported periodically by application
Goals of local scheduling policies

- Maximize rate of granted credit
  - utilize CPU
  - avoid missed deadlines
- Enforce resource shares over long term
  - don’t count long unavailable periods
- Maximize project variety
  - avoid running 1 project for weeks
## Scheduling scenarios

<table>
<thead>
<tr>
<th>Project</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource share</td>
<td>0.4</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>Job length hours</td>
<td>100</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>Job deadline hours</td>
<td>200</td>
<td>12</td>
<td>5</td>
</tr>
</tbody>
</table>

- **ConnInterval:** 12 hrs
- **MaxCPUS:** 1
Early scheduling policies

- CPU scheduling: round-robin, weighted by resource share
- Work fetch: maintain at least ConnInterval work, proportional to resource share, at least one job per project
- This fail disastrously for the given scenario (all deadlines missed)
- Earliest deadline first: a little better, but most deadlines missed
Current scheduling policies

- Job completion estimation
  - maintain “duration correction factor” estimate
    
    estimate: $FA + (1-F)B$
    
    where $F$ is fraction done,
    
    $A$ is estimate based on CPU time and $F$,
    
    $B$ is estimate based on FP count and benchmarks

- Debt (per project)
  - short-term: used for CPU scheduling
  - long-term: used for work fetch
Round-robin simulation

- Computes: deadline misses; per-project CPU shortfall; total CPU shortfall
- Example: 2 CPUs; project A has share 2 and jobs A1, A2; project B has share 1 and job B1
CPU scheduling policy

- Do round-robin simulation
- EDF among projects with deadline misses
- weighted round-robin among others
- avoid preempting jobs that haven’t checkpointed recently
Work fetch policy

- A project is “overworked” is
  \[ \text{LongTermDebt}(P) < -\text{SchedulingInterval} \]
  (this happens if P’s jobs have tight deadlines and are usually run in EDF mode)

- Policy: find a project P that maximizes
  \[ \text{LongTermDebt}(P) + \text{Shortfall}(P) \]
  and is not overworked, and has no deadline misses
- Ask it for \( \max(\text{Shortfall}(P), \frac{\text{TotalShortfall}}{\text{share}}) \) work
Memory-aware scheduling

- Periodically measure working-set size of running applications, maintain smoothed average
- Run only a set of jobs that fits in allowed RAM
Future work

- More intelligence in server
  - scheduler RPC includes list of queued jobs, including deadlines and fraction done
  - scheduler runs EDF simulation to see if each prospective job will jeopardize deadlines
- Maintain better statistical info on
  - job run-times
  - availability, network connectivity info
- Develop simulator to evaluate policies