Grid Computing

Exposing the myths of desktop scavenging grids

John Easton – IBM Grid computing
JKJ@uk.ibm.com

With so much "compute power" out there…

SETI@Home

- Sat 08 Oct 21:08:37 2005 GMT
- URL: setiathome.ssl.berkeley.edu/

- Total credit granted:
  - 2,799,193,542 cobblestones
- Total users:
  - 220,769 users
- Total hosts:
  - 469,244 host(s)

World Community Grid

- 10/09/2005 18:06:02 (UTC)
- URL: www.worldcommunitygrid.org

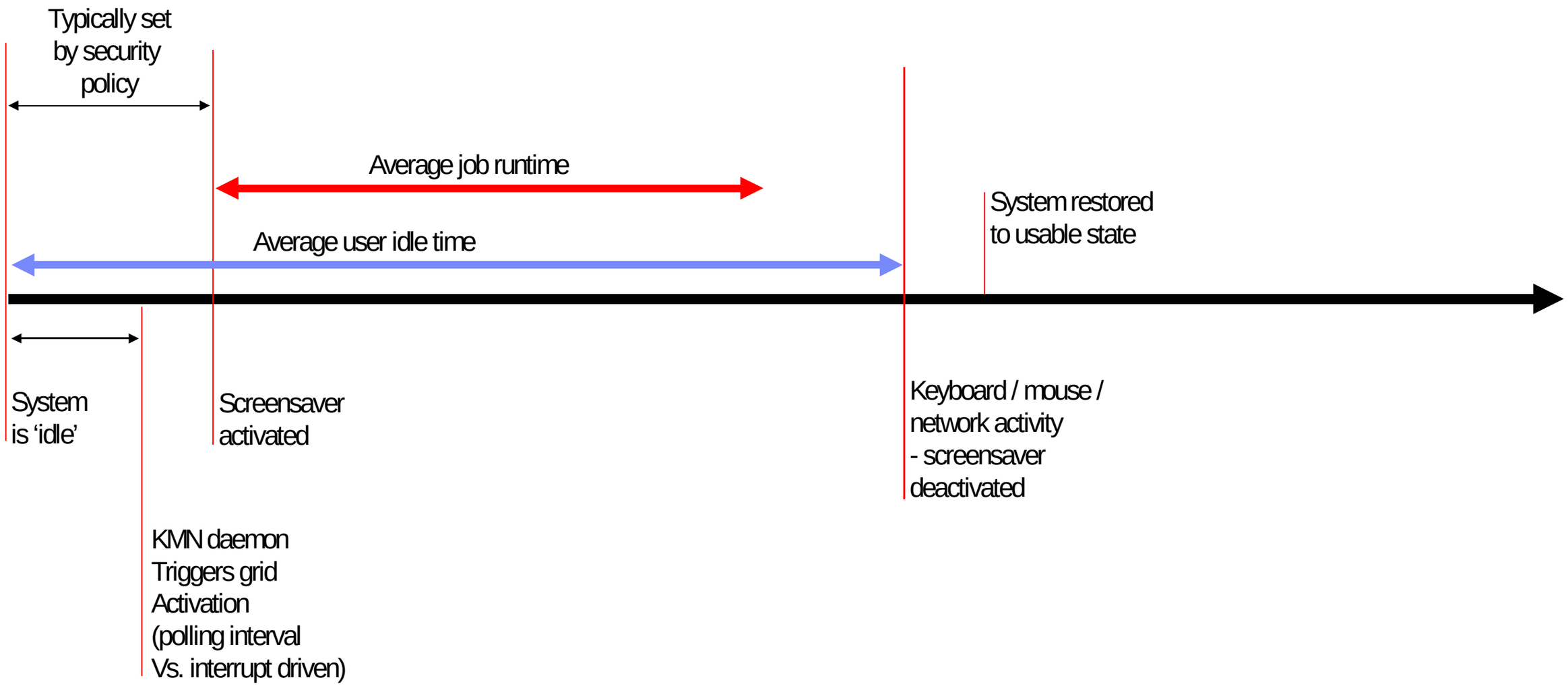
- Total Run Time (y:d:h:m:s)
  - 17,675:218:05:11:19
- Members
  - 94,038
- Devices
  - 152,071

Cobblestone 1, is 1/100 day of CPU time on a reference computer that does:
1,000 double-precision MIPS based on the Whetstone benchmark.
1,000 VAX MIPS based on the Dhrystone benchmark.

On this basis 2799193542 cobblestones = 76,690 years of 1000 VAX MIPS

…how come so many companies have failed to make desktop scavenging "work"?

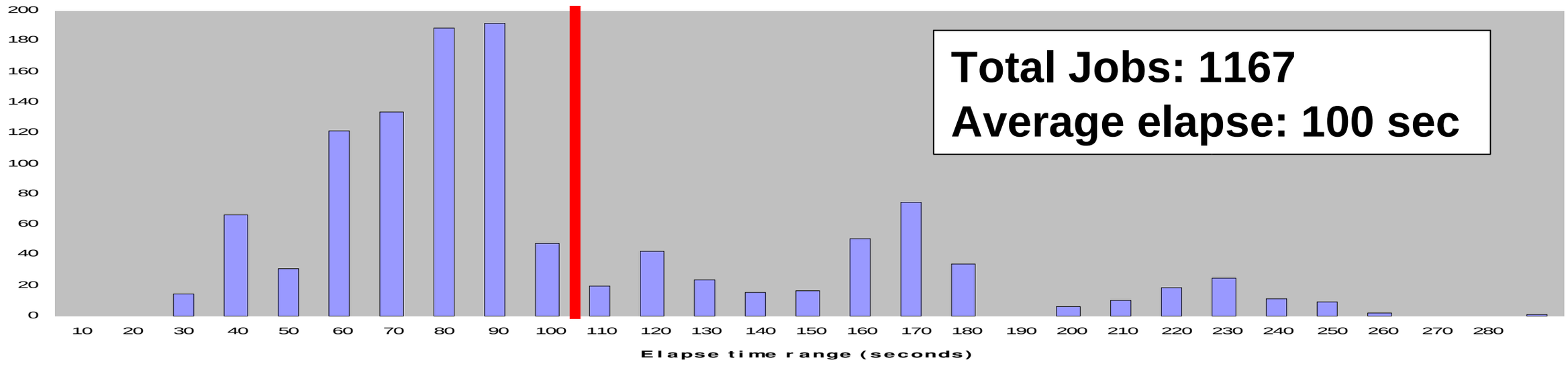
A desktop scavenging “timeline”

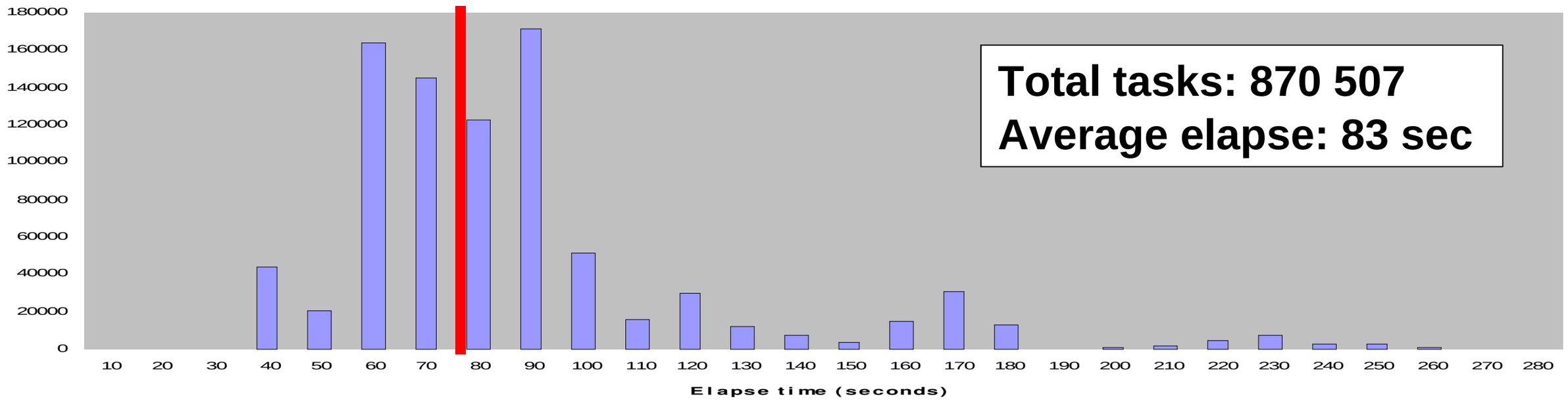


Things to think about….

- Push vs. pull scheduling
- 100% utilisation is generally a BAD thing for a PC
- How do you know the machine is available for work?
  - Screensaver kicks in
  - KMN (keyboard / mouse / network) activity
    - Polling vs. interrupt-driven
- Job criticality
  - Response time
- What is the critical mass of systems to deliver response time AND throughput?
  - System population vs. user activity profiles vs. application considerations

Job & task elapse distributions

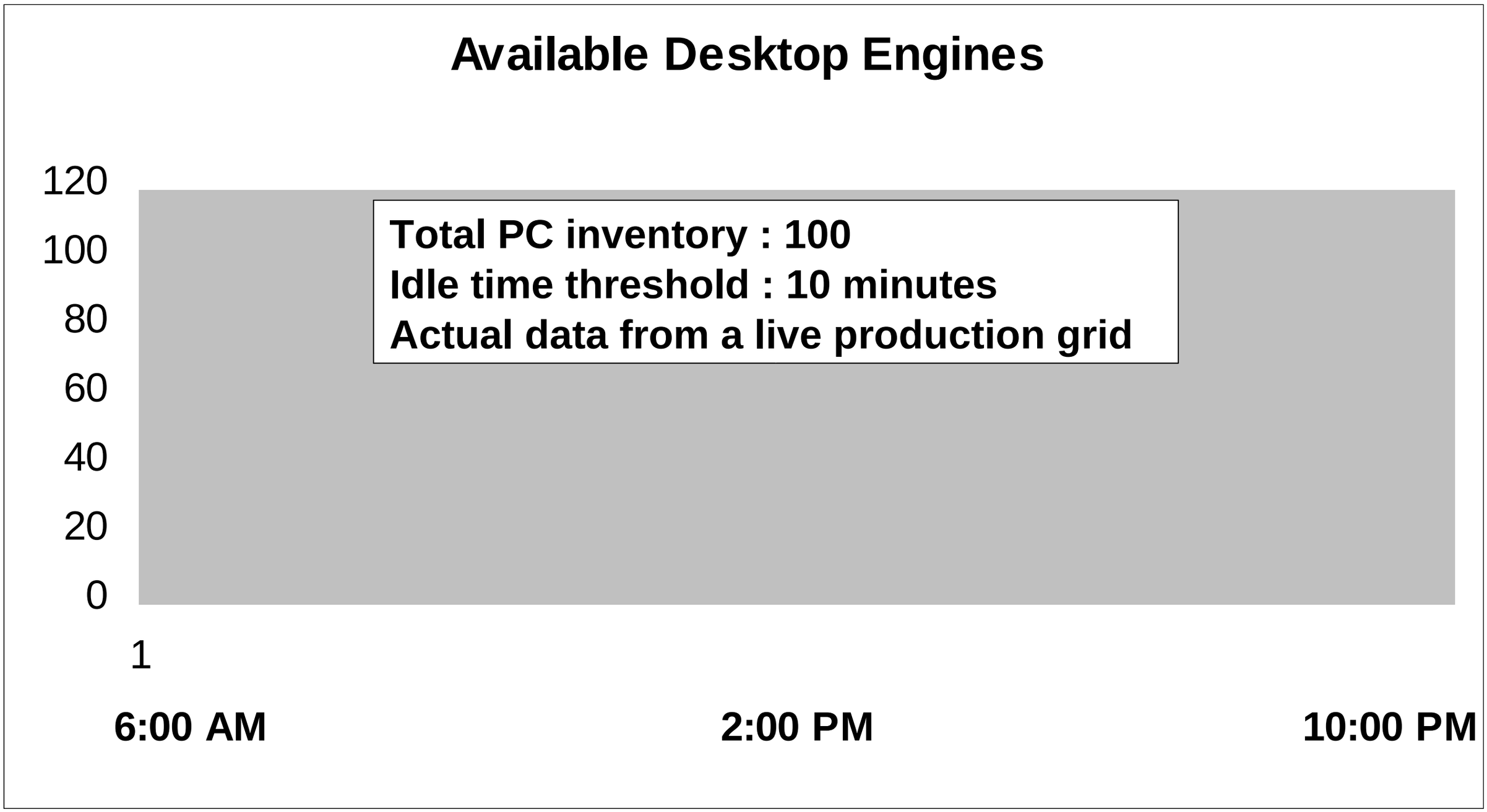




Problem statement

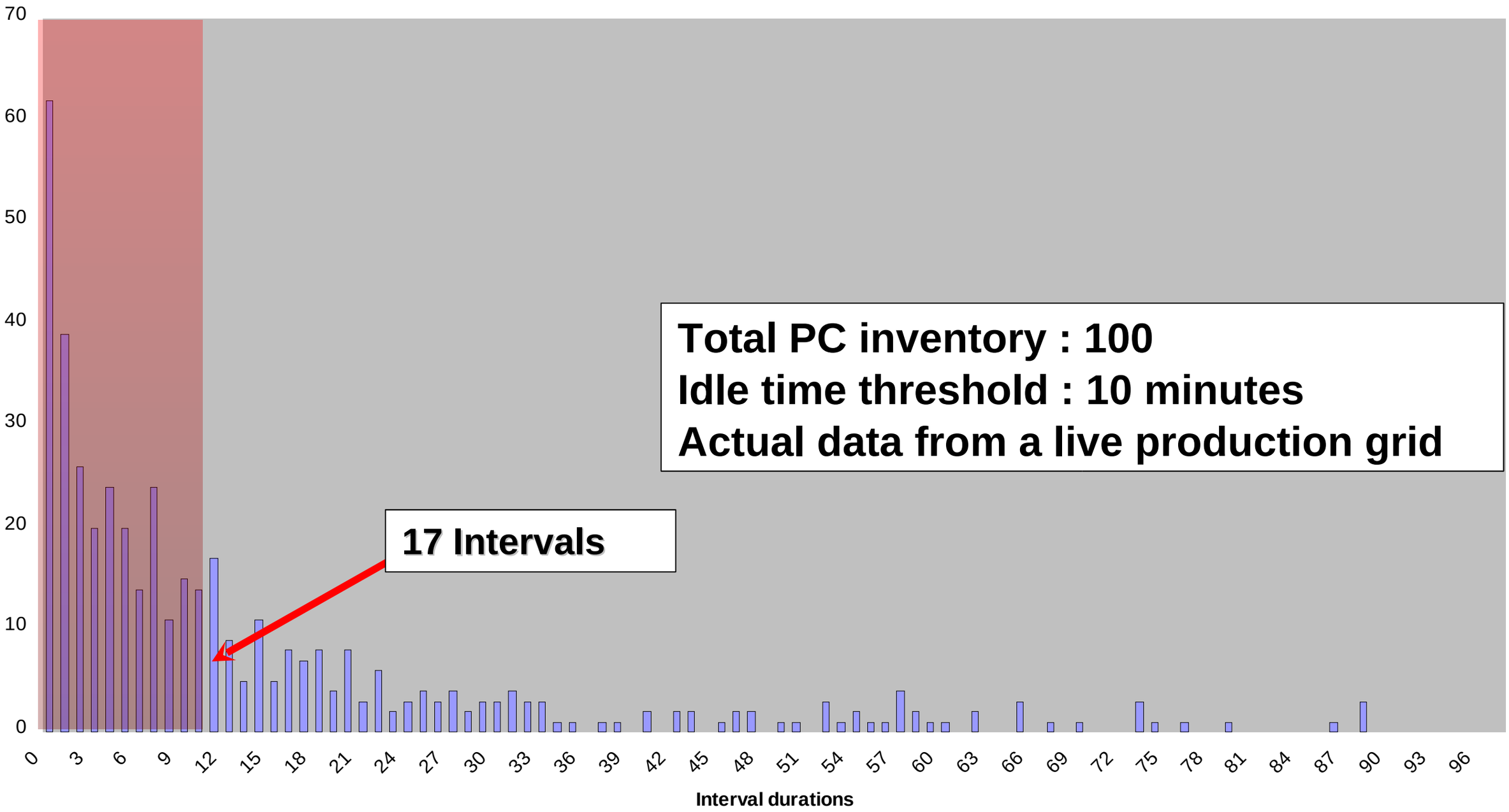
- Average task duration = 83 seconds on a 1.7 GHz CPU.
  - 2.5 minutes to complete on the standard customer configuration of a 800 MHz Intel.
- To determine whether tasks lasting on average 2.5 minutes would statistically complete on a PC grid, during the day time, in intervals where PC users would be inactive on their machine.
  - How often will an active task be kicked out by a user getting back to his machine.
- The answer to this question depends on two major data.
  - Processing priority in favor of the total restitution time
    - Redispatch the task to another engine in case of user activity vs. a mechanism that would put the process into sleep mode for an unpredictable duration
  - What is the statistical behavior of the average PC user?
    - How many intervals is he/she is away from the PC during the day
    - What are durations of those intervals.
- Strategy used by the grid to determine when an interval should start.
  - The usual strategy is to assume that if the user has been inactive for say 10 minutes, the chance that they will return in the next minute is low enough to consider sending a task to the PC.

PC Grid availability daily profile



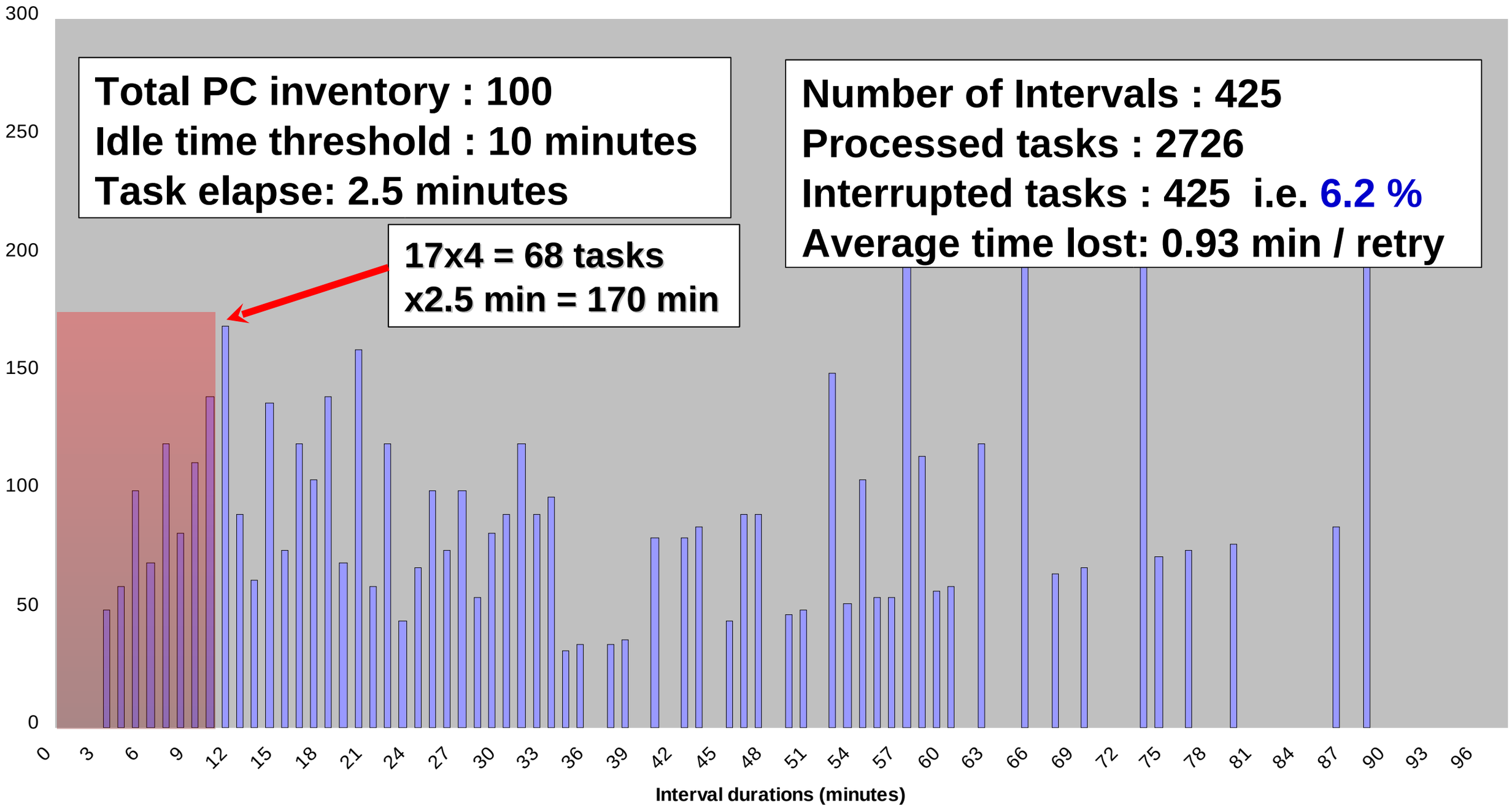
- On average only 20% of systems are available during the working day !!
- This is for ‘desktop’ systems – imagine what the effect of laptops is!!

PC Grid usability analysis



-If we assume a 10 minute idle time and a 2.5 minute runtime there are 17
-intervals of between 11 and 12 minutes

PC Grid Usability Simulation



Results

- 17 intervals of between 11 & 12 minutes can accommodate on average 4 tasks of 2.5 minutes each.
  - 17 intervals could support processing of 68 tasks (4 x 17) => 170 minutes of usable capacity with 17 interrupted tasks.
- Analysis across all possible intervals gives cumulated figures of 425 intervals supporting up to 2726 task executions with only 425 interruptions.
  - A 6.2 % probability of interruption and an average lost time of less than 1 minute
  - 2726 intervals represent 14% of the total PC capacity across an 8 hours window.
- This kind of analysis demonstrated that the Grid was indeed viable for this organisation.
- The target deployment was 1000 PCs delivering an average of 140 PCs full in addition to the 1000 PCs capacity fully available during nights and week ends.
- If average task duration had been longer, then the probability for each task to be interrupted would have been higher.
  - For longer running tasks, this grid implementation would created significant network traffic without any work done. Results would have been unacceptable.

Conclusions

- 100% utilisation is generally a bad thing on a PC
  - It is better to run on an idle system than ‘compete’ with the user
- Criticality of response time?
  - Task runtime should be shorter than average idle time
  - How do you work out whether the dispatch a work-unit to an idle PC?
- What is the critical mass of systems to deliver response time AND throughput?
  - System population vs. user activity profiles vs. application considerations

IBM

Questions?