CSC 2228 Project

Dynamic Voltage Scaling in Mobile Devices

David Tam, Winnie Tsang, Catalin Drula

What We Did?

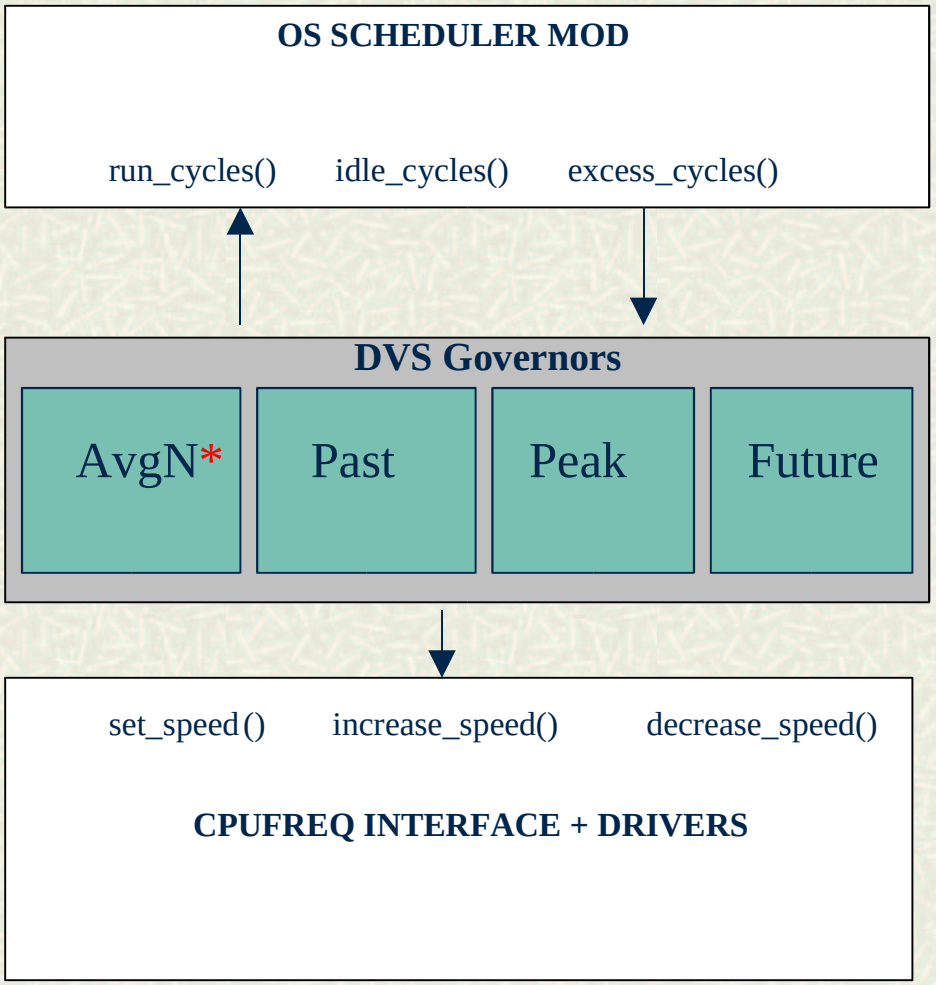
- Implement DVS algs
  + develop our own
- Target: Linux 2.6 & Intel P4-M

Why is this important?

- DVS goal = save energy
- Explore DVS algs on Linux 2.6 & P4-M SpeedStep
- Used hypothetical idle times
- Developed new DVS alg with OS info
- DVS alg:
  - AvgN*, PAST, PEAK, FUTURE

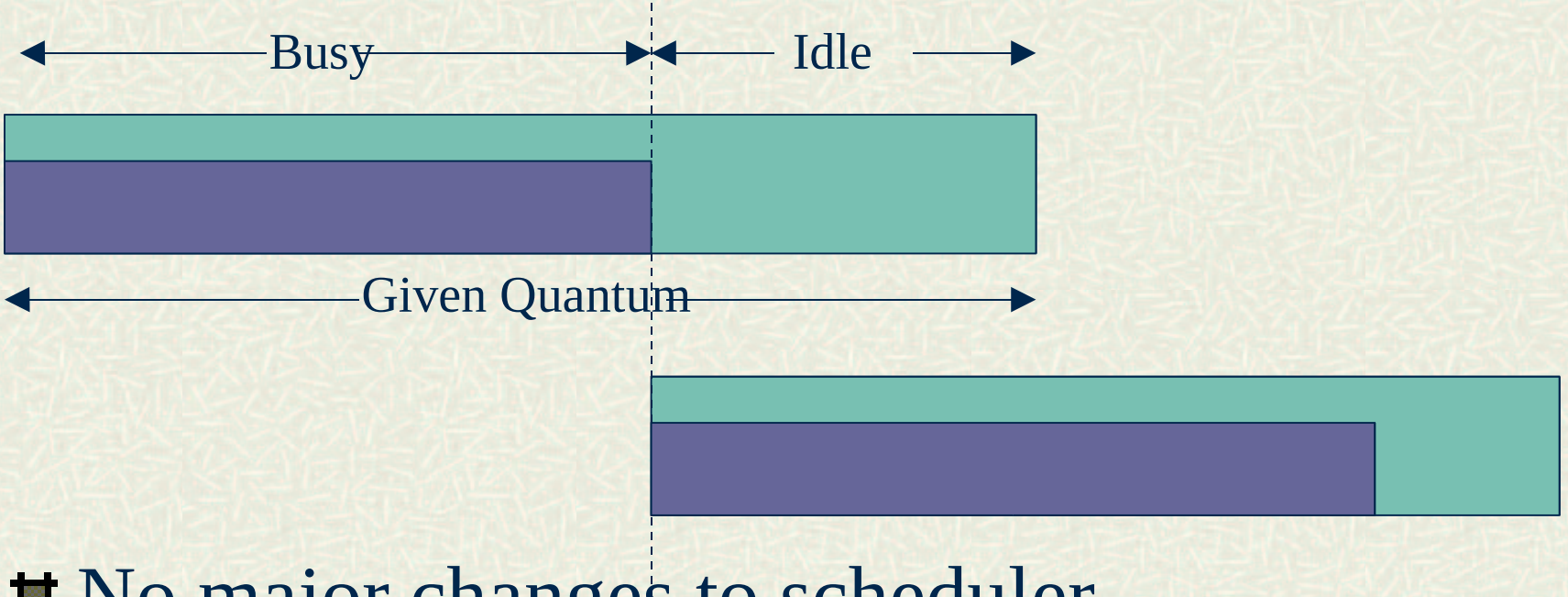
Architecture

3 Components:



DVS Interpretation

- Goal: stretch task to consume 99% of the given quantum



- No major changes to scheduler

DVS ALGORITHMS

- **FUTURE**
  - Our own proposed algorithm
  - Idea:
    - Look into the run_queue:
      - Get hypothetical idle time in the future
  - Prediction and Speed Setting is same as PAST

EXPERIMENTAL SETUP

Platform:

- IBM Thinkpad T30
- Redhat 9 2.6.0-test9 Linux kernel.
- Intel Mobile Pentium 4 - M processor

Frequency (GHz)	Voltage (V)
1.6	1.3
1.2	1.2
Idle-state	-

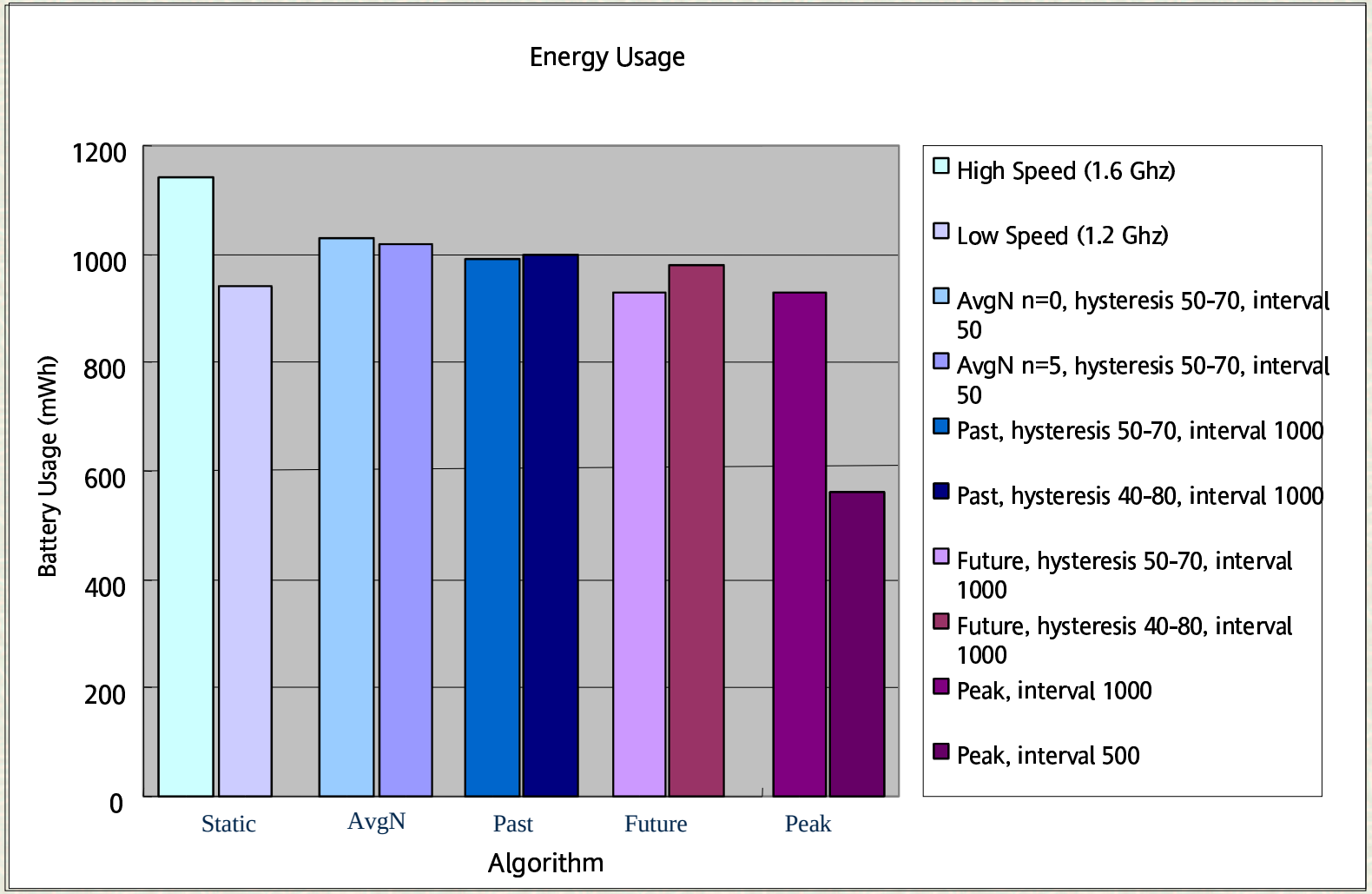
- Used ACPI in Linux to measure capacity (mWh)

EXPERIMENTAL SETUP

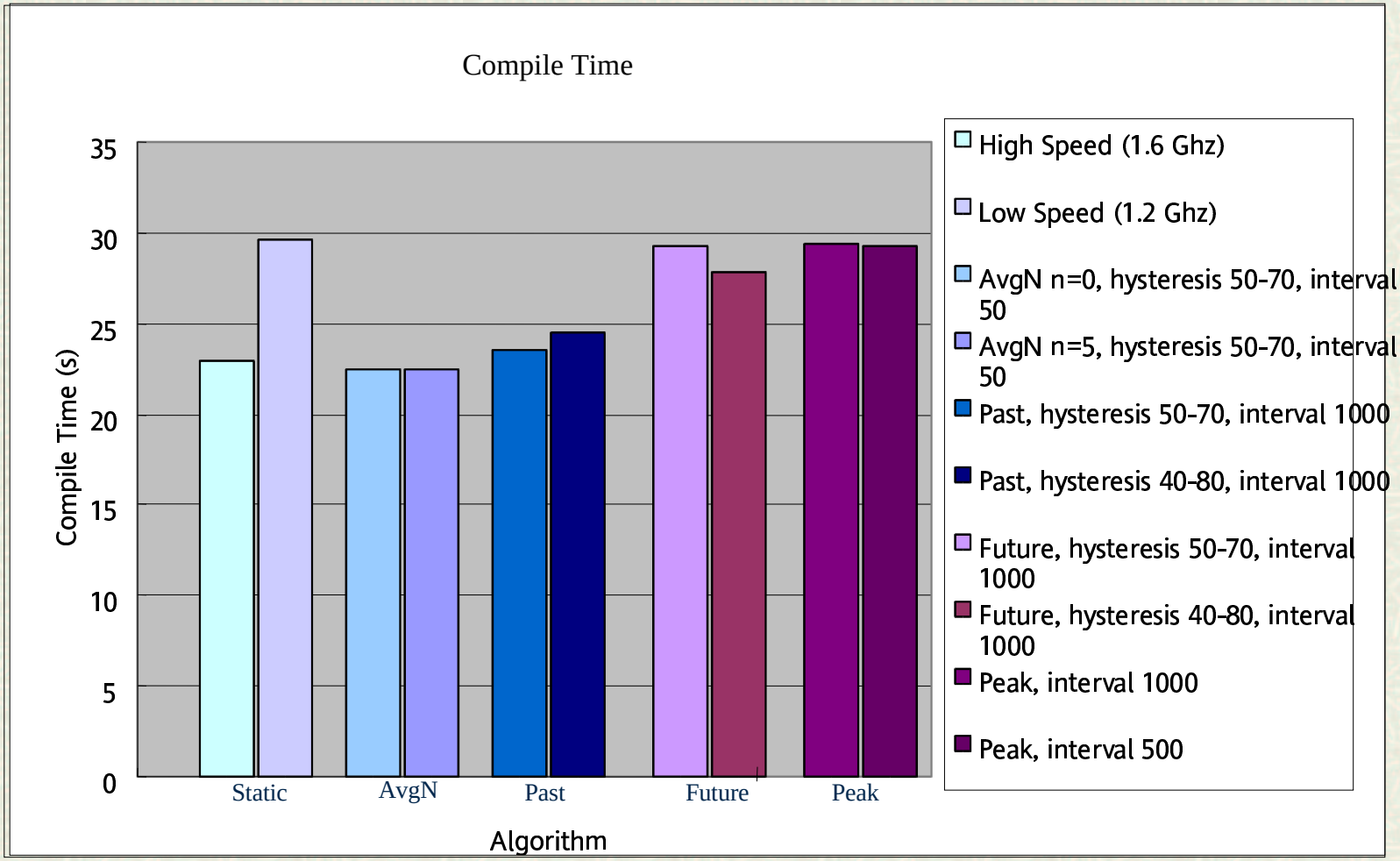
Experiments:

- Workload is 2m30sec long and consists of:
  - An mpeg video running in the background
  - Sleep for 20 seconds
  - Compile Linux kernel (linking stage)
    - lasts 20-30 seconds
  - Sleep for 20 seconds
  - Same compilation again

EXPERIMENTAL RESULTS



EXPERIMENTAL RESULTS



FUTURE WORK

- Implementation of excess cycle
- Fine tuning DVS alg parameters
- More workloads
- New DVS algs

CONCLUSION & CONTRIBUTIONS

- Best policies are AVGn and PAST
- DVS algs on new OS & real hardware
  - Used hype idle times
- “ Future” alg
- Contributions to Linux community:
  - 4 governors for CPUFreq
  - Make available on web