Notes on Multicore Thermal Management

**Problem**
- As processors get faster and more complicated, they often produce more heat
- Heat has been the biggest byproduct of power helping to make the “Power Wall”
- Localized hotspots can limit the performance of the entire processor

**New Approach**
- Considers several hardware and OS DTM (Dynamic Thermal Management) techniques
- Breaks techniques into 3 orthogonal axes and tries all combinations
  - Low-level control policies to reduce power
    - Stop-go - when processor is too hot (above 84.2°C) stop it for 30ms to cool down
    - DVFS (Dynamic Voltage and Frequency Scaling)
      - By turning down the clock rate, the voltage can also be lowered and thus significantly reduce power
      - On Intel dual core, a 20% slowdown in clock cuts the power use in half
    - Uses a PI-controller to dynamically adjust
  - Granularity to apply policy
    - Global - applies policy to all cores at once (so if one is hot, all affected)
    - Distributed - applies policy on a per-core basis
  - Migration policy - OS can move threads between cores to try to balance heat
    - Never move
    - Counter-based - count resource usage per unit time and approximate heat
    - Sensor-based - measure temperature of resources to get heat
- Experiment is conducted by running SPEC CPU programs in a software simulation
  - Uses Turandot and PowerTimer to model heat of 4-core PowerPC at 3.6 GHz

**Results**
- Performance measured by duty-cycle metric
  - Processor working full time at full speed is 100%
  - If turned off by stop-go or doing OS overhead for DTM, don’t count as work
  - If scaled down for DVFS scale duty-cycle (80% clock speed is 80% duty-cycle)
- Performance improvements all relative to distributed stop-go with no migration
- DVFS yields single biggest improvement of any of the techniques (2x improvement)
- Applying policy distributed helped significantly (~+.5x improvement)
  - Non-trivial to do for DVFS because of clocking, but probably worth it
- Sensor-based migration slightly better than counter-based
  - Both migration techniques help little for DVFS (~+.1x improvement)
  - Migration is a big help for stop-go (2x improvement)
- Distributed DVFS performs best with 2.5x improvement (2.6x with migration)
- All results shown in Table 8 (pg. 9)

**Why This is Different Than Other Work**
- Much work has been done focussing on overall power use without attacking hotspots
  - A low power design can have bad hotspots limiting performance
- Other research has handled static heat issues
- Other research has tried a few techniques, but hasn’t tried all combinations

**Lacking**
- The effects of SMT, heterogeneous multicore, or > 4 programs
- Programs run are regular SPEC CPU, what about server/datacenter programs?
- How hard is it to implement distributed DVFS?