Dynamic Thermal Management for Distributed Systems

Andreas Weissel • Frank Bellosa
Department of Computer Science 4 (Operating Systems)
University of Erlangen-Nuremberg
Martensstr. 1, 91058 Erlangen, Germany
{weissel,bellosa}@cs.fau.de
Benefits of Dynamic Thermal Management

- Cooling servers, server clusters
  - cooling facilities often dimensioned for worst-case temperatures or overprovisioned

- Guarantee temperature limits
  - no need for overprovisioning of cooling units
  - reduced costs (floor space, energy consumption, maintenance, ...)

- Increased reliability
  - safe operation in case of cooling unit failure
  - avoid local hot-spots in the server room

- Temperature sensors
Drawbacks of Existing Approaches

- If critical temperature is reached
  - throttle the CPU:
    - e.g. halt cycles, reduced duty cycle, reduced speed

- But: neglect of application-, user- or service-specific requirements due to missing online information about
  - the originator of a specific hardware activation and
  - the amount of energy consumed by that activity

→ Throttling penalizes all tasks
Outline

- From events to energy
  - event-monitoring counters
  - on-line estimation of energy consumption
- From energy to temperature
  - temperature model
- *Energy Containers*
  - accounting of energy consumption
  - task-specific temperature management
- Infrastructure for temperature management in distributed systems
Approaches to Energy Characterization

- Reading of thermal diode embedded in modern CPUs
  - low temporal resolution
  - significant overhead
  - no information about originator of power consumption
Approaches to Energy Characterization

- Reading of thermal diode embedded in modern CPUs
  - low temporal resolution
  - significant overhead
  - no information about originator of power consumption

- Counting CPU cycles
  - time as an indicator for energy consumption
  - time as an indicator for contribution to temperature level
  - throttling according to runtime
  - but: wide variation of the active power consumption
Approaches to Energy Characterization

- P4 (2 GHz) running compute intensive tasks: CPU load of 100%
  - variation between 30–51 W
From Events to Energy: Event-Monitoring Counters

- Event counters register energy-critical events in the complete system architecture.
  - several events can be counted simultaneously
  - low algorithmic overhead
  - high temporal resolution
  - fast response

- Energy estimation
  - correlate a processor-internal event to an amount of energy
  - select several events and use a linear combination of these event counts to compute the energy consumption

\[
\text{Energy} = \sum_{i} \text{#event}_i \cdot \text{weight}_i
\]
From Events to Energy: Methodology

- Measure the energy consumption of training applications
- Find the events with the highest correlation to energy consumption
- Compute weights from linear combination of event counts and real power measurements of the CPU

→ solve linear optimization problem:
find the linear combination of these events that produce the minimum estimation error

\[ \min \left\| \sum \#\text{event}_i \cdot \text{weight}_i - \text{measured energy} \right\| \]

→ avoid underestimation of energy consumption

\[ \text{measured energy} \leq \sum \#\text{event}_i \cdot \text{weight}_i \]
From Events to Energy: Methodology

- Set of events and their weights

<table>
<thead>
<tr>
<th>event</th>
<th>weight [nJ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>time stamp counter</td>
<td>6.17</td>
</tr>
<tr>
<td>unhalted cycles</td>
<td>7.12</td>
</tr>
<tr>
<td>µop queue writes</td>
<td>4.75</td>
</tr>
<tr>
<td>retired branches</td>
<td>0.56</td>
</tr>
<tr>
<td>mispred branches</td>
<td>340.46</td>
</tr>
<tr>
<td>mem retired</td>
<td>1.73</td>
</tr>
<tr>
<td>ld miss 1L retired</td>
<td>13.55</td>
</tr>
</tbody>
</table>

- Limitations of the Pentium 4
  - insufficient events for MMX, SSE & floating point instructions
  - the case for dedicated Energy Monitoring Counters
Outline

- From events to energy
  - event-monitoring counters
  - on-line estimation of energy consumption
- From energy to temperature
  - temperature model
- Energy Containers
  - accounting of energy consumption
  - task-specific temperature management
- Infrastructure for temperature management in distributed systems
From Energy to Temperature: Thermal Model

- CPU and heat sink treated as a black box with energy in- and output

- energy input: electrical energy being consumed
- energy output: heat radiation and convection
From Energy to Temperature: Thermal Model

- **Energy input**: energy consumed by the processor

\[ dT = c_1 P dt \]

- \( P \): CPU power consumption

![Diagram showing energy input, CPU, thermal sink, and energy output]

Energy\(#\text{Events}\) \rightarrow \text{CPU} \rightarrow \text{Energy}(\Delta\text{Temp})
From Energy to Temperature: Thermal Model

- **Energy output**: primarily due to convection

\[
\frac{dT}{dt} = c_1 P dt
\]

- **Energy output** (primarily due to convection)

\[
\frac{dT}{dt} = -c_2 (T - T_0) dt
\]

- **T** : ambient temperature

**Symbols**:
- \( P \): CPU power consumption
- \( T \): temperature
- \( T_0 \): ambient temperature
- \( c_1 \), \( c_2 \): constants

**Diagram**:
- Energy (#Events) → Heat Sink (Thermal Resistance) → CPU (Thermal Capacity) → Energy (ΔTemp)
From Energy to Temperature: Thermal Model

Altogether:
\[ dT = [c_1 P - c_2 (T - T_0)] dt \]

- energy estimator \(\rightarrow\) power consumption \(P\)
- time stamp counter \(\rightarrow\) time interval \(dt\)
- the constants \(c_1, c_2\) and \(T_0\) have to be determined
From Energy to Temperature: Thermal Model

- Altogether:
  \[ dT = [c_1 P - c_2(T - T_0)] dt \]
  - energy estimator \(\rightarrow\) power consumption \(P\)
  - time stamp counter \(\rightarrow\) time interval \(dt\)
  - the constants \(c_1, c_2\) and \(T_0\) have to be determined

- Solving this differential equation yields
  \[
  T(t) = \frac{-c_0}{c_2} \cdot e^{-c_2 t} + \frac{c_1}{c_2} \cdot P + T_0
  \]
  \[
  \frac{-c_0}{c_2} \cdot e^{-c_2 t} \quad \text{dynamic part}
  \]
  \[
  \frac{c_1}{c_2} \cdot P + T_0 \quad \text{static part}
  \]
Thermal Model: Dynamic Part

- Measurements of the processor temperature
  - on a sudden constant power consumption and
  - a sudden power reduction to HLT power.
- fit an exponential function to the data: coefficient = $c_2$

![Graph showing temperature over time for different states: Idle (50 W) and Active (12 W)]
Thermal Model: Static Part

- Static temperatures and power consumption of the test programs
Thermal Model: Static Part

- Linear function to determine $c_1$ and $T_0$
Thermal Model: Implementation

- Linux 2.6 kernel
- Periodically compute a temperature estimation from the estimated energy consumption
- Deviation of a few degrees celsius over 24 hours
  - or if ambient temperature changes
- Re-calibration with measured temperature every few minutes
Thermal Model: Accuracy

![Graph showing estimated vs. measured temperature over time](image-url)
Outline

■ From events to energy
  ◆ event-monitoring counters
  ◆ on-line estimation of energy consumption
■ From energy to temperature
  ◆ temperature model
■ Energy Containers
  ◆ accounting of energy consumption
  ◆ task-specific temperature management
■ Infrastructure for temperature management in distributed systems
Properties of Energy Accounting

- Accounting to different tasks/activities/clients
  - example: web server serving requests from different client classes
  - e.g. Internet/Intranet, different service contracts
- “Resource principal” can change dynamically
- Client/server relationships between processes
  - account energy consumption of server to client
Energy Containers

- Resource Containers [OSDI '99] ➔ Energy Containers
  - separation of protection domain and “resource principal”

- Container Hierarchy
  - root container (whole system)
  - processes are attached to containers
  - this association can be changed dynamically (client/server relationship)
  ➔ energy is automatically accounted to the activity responsible for it

- Energy shares
  - amount of energy available (depending on energy limit)
  - periodically refreshed
  - if a container runs out of energy, its processes are stopped
Energy Containers

- Example:
  web server working for two clients with different shares

![Graph showing power consumption over time](image-url)

- Start throttling
Task-specific Temperature Management

- Periodically compute an energy limit for the root container (depending on the temperature limit $T_{\text{limit}}$)
  \[dT = [c_1P + c_2(T - T_0)]dt \leq T_{\text{limit}} - T\]

- Dissolve to $P \rightarrow P_{\text{limit}}$

- Energy budgets of all containers are limited according to their shares

- Tasks are automatically throttled according to their contribution to the current temperature

- Throttling is implemented by removing tasks from the runqueue
Temperature Management

Example: Enforcing a temperature limit of 45°C
Outline

- From events to energy
  - event-monitoring counters
  - on-line estimation of energy consumption
- From energy to temperature
  - temperature model
- Energy containers
  - accounting of energy consumption
  - task-specific temperature management
- Infrastructure for temperature management in distributed systems
Energy Containers

- Distributed energy accounting

- Id transmitted with the network packets (IPv6 extension headers)
- Receiving process attached to the corresponding energy container
- Temperature and energy are cluster-wide accounted and limited
- Transparent to applications and unmodified operating systems
Energy Containers

- Energy accounting across machine boundaries
  - requests from two different clients represented by two containers
  - web server sends requests to factorization server

The energy consumption of the server is correctly accounted to the client.
Infrastructure for DTM in Distributed Systems

- Distributed energy accounting
- Foundation for policies managing energy and temperature in server clusters
  - account, monitor and limit energy consumption and temperature of each node
- Examples
  - set equal energy/temperature limits for all servers
    - cluster-wide uniform temperature and power densities, no hot spots in the server room
  - use energy/temperature limits to
    - throttle affected servers in case of a cooling unit failure
    - reduce number of active cooling units in case of low utilization
Conclusion

- Event-monitoring counters enable
  - on-line energy accounting
  - task-specific temperature management
- Correctly account client/server relations across machine boundaries
- Transparent to applications and unmodified operating systems

Future directions
- examine more sophisticated energy models
- task-specific frequency scaling to adjust the thermal load