SYNERGISTIC DESIGN OF ENERGY-EFFICIENT HETEROGENEOUS COMPUTE NODES

Abdoulaye Gamatié
LIRMM / CNRS-UM, Montpellier

Colloque National du GDR SOC2, June 2017, Bordeaux

Nowadays technological ecosystem

- Internet of Things
- Cloud
- Data center
- Supercomputers
- Embedded
- Energy-efficiency
FOCUS ON SUPERCOMPUTING DOMAIN

The quest of Exascale Supercomputers
Scientific challenges for HPC: some applications

**Hurricane prediction**
Avoid thousands human victims, e.g. Katrina, Sandy…

**Predictive calculation in astrophysics**
Understanding the formation and evolution of the universe

**Combustion simulation**
Low temperature combustion for efficient and low emission automobile engines

**Simulation of cells in biology**
Advance comparative genomics and underlying phylogenics, cell degradation
Goal: exascale supercomputers

$10^{18}$ floating point operations per second (FLOPs)

**Target power: maximum 20 MW around 2020**

**Today:**

Sunway TaihuLight

- $93 \times 10^{15}$ FLOPS
- $10.65 \times 10^6$ cores
- $1.3 \times 10^6$ GB memory
- $246$ M€ cost

- $15.4$ MW
- $1155$ €/h ($10$ M€/y)

With current technology ➔ 3ExaFLOPs is 1GW budget
Funny facts: towards reduction of energy cost

World's first Petaflop supercomputer Roadrunner is shut down

Was using too much power
By Lee Bell
Tue Apr 02 2013, 11:03

THE WORLD'S FIRST Petaflops supercomputer Roadrunner has been switched off after five years of operation due to its high energy consumption.

The IBM built supercomputer cluster, which is housed at the US Los Alamos National Laboratory, was the first to break the Petaflops barrier of one quadrillion calculations per second when it was launched in 2008.
Funny facts: towards reduction of energy cost (cont’d)

Microsoft research project puts cloud in ocean for the first time
FOCUS ON EMBEDDED DOMAIN

Smarter world at minimized energy budget
Trends in embedded computing domain

- Continuous integration of new functionalities

• Increasing demand in performance: on-chip multicore systems
Increasing performance demand: some numbers

**Automotive**

- Source: A. Grasset (Thales)

**Space**

- Source: NASA, 2009

**Avionics**

- Source: A. Grasset (Thales)

---

[Goto, 2013]

[Davey, 2013]

[Ladier, 2008]
The quest of energy-efficiency

Low power

Embedded computing

Heterogeneous computing

Services++

Functionalities++

=> Performance++

Performance

High-performance computing

- 93 x 10^{15} FLOPS
- 15.4 MW (1155 €/h (10 M€/y)
- 10.65 x 10^6 cores / 1.3 x 10^6 GB memory
- 246 M€ cost

Energy++ <= Exascale computing
Outline of this talk

• **Synergistic design of heterogeneous compute nodes**
  • Motivation and requirements

• **Examples of synergistic designs for energy-efficiency**
  • Adaptive multicore compute node design
  • Leveraging non volatile memory with compiler optimization

• Conclusion
Compute node design: quick brainstorming

Suitable programming models?
(languages, compilation, runtime...)

FLC → homogeneous memory hierarchy
LLC → heterogeneous memory hierarchies
Main →

homogeneous multicore
heterogenous multicore with accel.
Synergistic design

Key idea: dealing with multi-level design concerns

- **Requirement #1**: suitable integrated frameworks
  - design and evaluation

- **Requirement #2**: cross-domain interaction

![Diagram showing Hw Architecture and Hw Technology with Application / Software](image)
Example: data movement issue

- On 22nm CMOS technology*
  - Toggling a single bit: \(10^{-6} \text{ pJ}\)
  - Moving a single bit: \(1 \text{ pJ/mm}\) (1 mW/mm at 1 GHz)

- Synergistic optimization for energy-efficiency: more than 80% gain

\[ 18\text{pJ/F} \to 9\text{pJ/F} \]
\[ 77\text{pJ/F} \to 18\text{pJ/F} \]

Outline of this talk

• **Synergistic design of heterogeneous compute nodes**
  • Motivation and requirements

• **Examples of synergistic designs for energy-efficiency**
  • Adaptive multicore compute node design
  • Leveraging non volatile memory with compiler optimization

• **Conclusion**
DESIGN EVALUATION FRAMEWORK

For heterogeneous compute nodes
Synergistic design: which frameworks?

Manycore Architecture enerGy and Performance evaluation environment: **MAGPIE**

- **gem5**: quasi-cycle accurate simulator
  - IPs: cores (x86, ARM...), memory, interconnect...
  - can boot a complete linux OS
  - detailed microarchitecture details / performance statistics

- **McPAT**: area, power and timing modeling for CMOS, SOI technologies

- **NVSim**: area, power and energy estimator for non-volatile memory technologies

⚠️ accuracy does not come for free!

** [http://www.gem5.org](http://www.gem5.org)
**** [http://nvsim.org](http://nvsim.org)
Synergistic design: MAGPIE automated flow

- **Automation scripts: make user life easy!**
  - No tedious and error-prone manual statistics manipulation
  - Significant time and effort saving

- o Sw: workloads, OS
- o Hw: cores / interconnect / memory parameters...

Platform components calibration

Manycore system execution

Power, area estimation

Postprocessing for graphical renderings
ENERGY-EFFICIENT COMPUTE NODE DESIGN

Software / hardware design tradeoff: part 1
Example of heterogeneous compute node

Single-ISA multicore

Exynos 5422 chip (Odroid-XU3 board)
How to reach energy-efficiency?

CONTINUUM ANR project
- leveraging compilation in adaptive compute node design
- http://www.lirmm.fr/continuum-project

Application / Software

LLVM

Hw Architecture

Runtime monitoring

Hw Technology
Motivational example: matrix multiplication

```c
int main(int argc, char** argv){
    int M1, N1, M2;
    setCoreAllocation(_config);
    int** m1 = reaMatrix(argv[1], &M1, &N1);
    ...
    int** m2 = reaMatrix(argv[2], &N1, &M2);
    ...
    setCoreAllocation(_config);
    int** m3 = multMatrix(m1, m2, M1, N1, M2);
    ...
    setCoreAllocation(_config);
    printMatrix(m1);
    printMatrix(m2);
    printMatrix(m3);
}
```
Motivational example: design-time analysis

```c
int main(int argc, char** argv){
    int M1, N1, M2;
    setCoreAllocation(LOW-POWER); // IO-intensive
    int** m1 = reaMatrix(argv[1], &M1, &N1);
    ...
    int** m2 = reaMatrix(argv[2], &N1, &M2);
    ...
    setCoreAllocation(HIGH-PERF); // Compute-intensive
    int** m3 = multMatrix(m1, m2, M1, N1, M2);
    ...
    setCoreAllocation(...);
    printMatrix(m1);
    printMatrix(m2);
    printMatrix(m3);
}
```
Motivational example: runtime analysis

```c
int main(int argc, char** argv){
    int M1, N1, M2;
    setCoreAllocation (LOW-POWER); // IO-intensive
    int** m1 = reaMatrix(argv[1], &M1, &N1);
    ...
    int** m2 = reaMatrix(argv[2], &N1, &M2);
    ...
    setCoreAllocation (?); // Depends on M1, N1 & M2
    int** m3 = multMatrix(m1, m2, M1, N1, M2);
    ...
    setArchitecture(...);
    printMatrix(m1);
    printMatrix(m2);
    printMatrix(m3);
}
```
Which approach to consider?

<table>
<thead>
<tr>
<th></th>
<th><strong>Design-time</strong> (e.g., compiler)</th>
<th><strong>Runtime</strong> (e.g., OS)</th>
<th><strong>Hybrid adaptive</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>pros</strong></td>
<td>Leverage program static features at <strong>lower runtime cost</strong></td>
<td>Leverage <strong>unpredictable</strong> runtime behavior for better choice</td>
<td>Leverage <strong>advantages of both</strong> Static and Dynamic</td>
</tr>
</tbody>
</table>
| **cons**       | Cannot address unpredictable behavior: e.g., input-dependent or environment-dep. executions | - Overhead of runtime monitoring  
- Getting all suitable insights from program behavior not easy | Requires a careful combination of Static and Dynamic decisions |
Hybrid adaptive allocation approach
ENERGY-EFFICIENT COMPUTE NODE DESIGN

Software / hardware design tradeoff: part 2
Let us reconsider things...

Runtime breakdown

- a) LITTLE cluster (A7@200MHz)
- b) big cluster (A15@2GHz)
- c) HMP (A7@200MHz/A15@2GHz)
How to reach energy-efficiency?

**MontBlanc** EU projects
- leveraging programming models in heterogeneous compute node design
- [http://montblanc-project.eu](http://montblanc-project.eu)
big.LITTLE architecture: modeling with gem5

Coherent interconnect

Exynos 5422 chip

20% avg error
Towards big.LITTLE asymmetric architecture

Energy-efficient designs based on Rodinia kernels

<table>
<thead>
<tr>
<th></th>
<th>Cortex-A7 cluster</th>
<th>Cortex-A15 cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Count</td>
<td>Clock</td>
</tr>
<tr>
<td>Conf. 1</td>
<td>4</td>
<td>800 MHz</td>
</tr>
<tr>
<td>Conf. 2</td>
<td>4</td>
<td>800 MHz</td>
</tr>
<tr>
<td>Conf. 3</td>
<td>7</td>
<td>800 MHz</td>
</tr>
<tr>
<td>Conf. 4</td>
<td>7</td>
<td>800 MHz</td>
</tr>
<tr>
<td>Conf. 5</td>
<td>7</td>
<td>800 MHz</td>
</tr>
</tbody>
</table>

Variable gain => programming model matters?

Existing configurations
- SMP (cluster A7@800MHz)
- SMP (cluster A15@2GHz)
- HMP (clusters A7@800/A15@800MHz)

Proposed configurations (HMP)
- C1
- C2
- C3
- C4
- C5

Execution time (ms)

EtroS (mJ)

Upto 27% less with asymmetric design
Leveraging programming models

OpenMP 3.0 thread-based fork-join model
• Symmetric worker threads

OmpSs (or OpenMP 4.0) task-based model
• Design-time task criticality analysis
• Criticality-aware dynamic task scheduling

Task Dependency Graph

Critical task queue

Non-Critical task queue
Leveraging programming models (cont’d)

Example of empirical design evaluation (Cholesky)
Leveraging programming models (cont’d)

Example of empirical design evaluation (Cholesky)

• Architecture design impact
• Programming model impact

Best synergistic energy-efficient configuration
Insight #1

Energy efficiency is sensitive to careful combination of heterogeneous architecture configurations with programming models / runtime.
ENERGY-EFFICIENT COMPUTE NODE DESIGN
Exploiting emerging memory technologies
Emerging memory technologies

- Magnetic memories: STT-RAM...
  - Non volatility, high density, good endurance...
  - Low leackage => negligible static power

<table>
<thead>
<tr>
<th></th>
<th>SRAM</th>
<th>DRAM</th>
<th>Flash (NOR)</th>
<th>Flash (NAND)</th>
<th>FeRAM</th>
<th>MRAM</th>
<th>PRAM</th>
<th>RRAM</th>
<th>STT-RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Non-volatile</strong></td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Cell size (F²)</strong></td>
<td>50–120</td>
<td>6–10</td>
<td>10</td>
<td>5</td>
<td>15–34</td>
<td>16–40</td>
<td>6–12</td>
<td>6–10</td>
<td>6–20</td>
</tr>
<tr>
<td><strong>Read time (ns)</strong></td>
<td>1–100</td>
<td>30</td>
<td>10</td>
<td>50</td>
<td>20–80</td>
<td>3–20</td>
<td>20–50</td>
<td>10–50</td>
<td>2–20</td>
</tr>
<tr>
<td><strong>Write / Erase time (ns)</strong></td>
<td>1–100</td>
<td>15</td>
<td>1 μs / 10 ms</td>
<td>1 ms / 0.1 ms</td>
<td>50 / 50</td>
<td>3–20</td>
<td>60 / 120</td>
<td>10–50</td>
<td>2–20</td>
</tr>
<tr>
<td><strong>Endurance</strong></td>
<td>$10^{16}$</td>
<td>$10^{16}$</td>
<td>$10^5$</td>
<td>$10^5$</td>
<td>$10^{12}$</td>
<td>$&gt;10^{15}$</td>
<td>$10^8$</td>
<td>$10^8$</td>
<td>$&gt;10^{15}$</td>
</tr>
<tr>
<td><strong>Write power</strong></td>
<td>Low</td>
<td>Low</td>
<td>Very high</td>
<td>Very high</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Other power consumption</strong></td>
<td>Current leakage</td>
<td>Refresh current</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><strong>High voltage required</strong></td>
<td>No</td>
<td>3 V</td>
<td>6–8 V</td>
<td>16–20 V</td>
<td>2–3 V</td>
<td>3 V</td>
<td>1.5–3 V</td>
<td>1.5–3 V</td>
<td>&lt;1.5 V</td>
</tr>
</tbody>
</table>

Figure 1: Leakage power becomes a growing problem as demands for more performance and functionality drive chipmakers to nanometer-scale process nodes (Source: IBM).

Motivational example

```c
volatile int x = 0;
for (i = 0; i < N; i++) {
    x = 0;
}
```

<table>
<thead>
<tr>
<th>(N = 5.10^7)</th>
<th>Exec. time (ms)</th>
<th>Energy (mJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>full-SRAM L1 cache</strong></td>
<td>305.13</td>
<td>79.5</td>
</tr>
<tr>
<td><strong>full-STTRAM L1 cache</strong></td>
<td>305.31 (+0.06%)</td>
<td>67.8 (-14.7%)</td>
</tr>
</tbody>
</table>

Possible gain in dynamic power?

- focus on the **expensive writes** in STT-RAM

Negligible static power
How to reach energy-efficiency?

- Compiler optimization
- Non volatile memory in cache hierarchy
Silent Stores

volatile int x = 0;
for (i = 0; i<N; i++) {
    x = 0;
}

redudant (expensive) writes: silent stores

• Definition
  • a store is “silent” if it writes the value that is already present in memory location

• Studied in the early 2000’s
  • Kevin M. Lepak, Gordon B. Bell, and Mikko H Lipasti “Silent stores and store value locality”. In IEEE TC 50(11), 2001
  • For monoprocessor performance speedup, and multiprocessor bus traffic improvement
  • Addressed at hardware level
Silent store elimination

• Identify likely silent stores by profiling and replace...

  store @x = val

  load y = @val
cmp val, y
beq next
store @x = val
next: ...

• Code modification at IR level during code generation
Is silent store elimination always profitable?

- This transformation is beneficial iff:
  \[\text{Cost}_{\text{load}} + (1 - P_{\text{silent}}) \text{Cost}_{\text{store}} \leq \text{Cost}_{\text{store}}\]

- Example of profitability threshold evaluation

```c
volatile int x;
// assuming N > M
For(i = 0; i<N; i++) {
  y = 0;
  if (i>M)
    y = i;
  x = y; // store
}
```
Motivational example (cont’d)

```c
volatile int x = 0;
for (i = 0; i<N; i++) {
    x = 0;
}
```

<table>
<thead>
<tr>
<th></th>
<th>Exec. time (ms)</th>
<th>Energy (mJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>full-SRAM</td>
<td>305.13</td>
<td>79.5</td>
</tr>
<tr>
<td>full-STTRAM</td>
<td>305.31 (+0.06%)</td>
<td>67.8 (-14.7%)</td>
</tr>
<tr>
<td>full-STTRAM + opt</td>
<td>305.31 (+0.06%)</td>
<td>64.6 (-18.7%)</td>
</tr>
</tbody>
</table>

Synergistic energy improvement: technology + compilation

Negligible static power + Reduced dynamic power
Related work

• Hardware approach
  • **Early-write termination**: when new value is already in memory, a control signal is generated to switch off write circuit and terminate


• Combined hardware and software approach
  • **Data allocation/migration** to reduce the cost of NVM write activities in hybrid memory: write-intensive data blocks stored in SRAM banks, read-intensive data blocks stored in NVM banks

  - Q. Li et al. « MAC: migration-aware compilation for STT-RAM based hybrid cache in embedded systems », ISPLED’12
  - Q. Li et al. « Compiler-assisted preferred caching for embedded systems with STT-RAM based hybrid cache », LCTES’12
  - J. Hu et al. « Data Allocation Optimization for Hybrid ScratchPad Memory with SRAM and Non-volatile Memory », IEEE VLSI 2013
Insight #2

Low-leakage emerging memory technology can be leveraged with portable software-level optimizations for improved energy-efficiency.
Outline of this talk

- **Synergistic design of heterogeneous compute nodes**
  - Motivation and requirements

- **Examples of synergistic designs for energy-efficiency**
  - Adaptive multicore compute node design
  - Leveraging non volatile memory with compiler optimization

- **Conclusion**
Conclusion

- Energy-efficiency is a major challenge in compute node design

- Energy-efficiency opportunities are present across different design layers

- Synergistic design as a solution
  - Design frameworks
  - Cross-domain interaction

- **Open question:** How to deal with multi-level optimization composability?
SYNERGISTIC DESIGN OF ENERGY-EFFICIENT HETEROGENEOUS COMPUTE NODES

Abdoulaye Gamatié
LIRMM / CNRS-UM, Montpellier

Colloque National du GDR SOC2, June 2017, Bordeaux