Experiences on the characterization of parallel applications in embedded systems with Extrae/Paraver

Adrian Munera, Sara Royuela, Germán Llort, Estanislao Mercadal, Franck Wartel, Eduardo quiñones

49th International Conference on Parallel Processing (ICPP2020)  
17-20 August 2020, Edmonton, AB, Canada
Use of parallelism in embedded systems

- Demand for **high level of performance** in embedded systems.
- **Heterogeneity** introduces complexity to exploit **performance portability**.
- **Parallel programming models** are fundamental for **productivity**.

- **OpenMP** is an appropriate solution to leverage the potential of the architecture:
  - Provides **time-predictability** ¹
  - Shows delimited **correctness** guarantees ²

Analyzing parallelism in embedded systems

- Parallelism affects functional and non-functional behavior (time, energy, memory, etc.)

- Need to analyze the impact of parallelism on the functional (FR) and non-functional (NFR) requirements.

<table>
<thead>
<tr>
<th>Analysis tool domain</th>
<th>Parallel programming model</th>
<th>Performance</th>
<th>NFR</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPC</td>
<td>✅</td>
<td>✅</td>
<td>❌</td>
</tr>
<tr>
<td>Embedded</td>
<td>❌</td>
<td>✅</td>
<td>✅</td>
</tr>
</tbody>
</table>
# Analysis tools: classification

<table>
<thead>
<tr>
<th>Data gathering method</th>
<th>Data storage method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic measurements</strong></td>
<td></td>
</tr>
<tr>
<td>Easy to obtain</td>
<td>Produce a summary of the picture</td>
</tr>
<tr>
<td>Come without information about factors</td>
<td>Lack information for specific points in time</td>
</tr>
<tr>
<td><strong>Sampling</strong></td>
<td></td>
</tr>
<tr>
<td>Provide better understanding of the application</td>
<td>Cannot characterize fine-grained tasks</td>
</tr>
<tr>
<td><strong>Instrumentation</strong></td>
<td></td>
</tr>
<tr>
<td>Captures the activity as it is</td>
<td>May introduce overhead</td>
</tr>
<tr>
<td>Come without information about factors</td>
<td></td>
</tr>
</tbody>
</table>

- Sampling: Provide better understanding of the application, Cannot characterize fine-grained tasks
- Instrumentation: Captures the activity as it is, May introduce overhead
- Profiling: Produce a summary of the picture
- Tracing: Capture exact picture, May introduce overhead
Analysis tools: from embedded to HPC systems

**EC**

- **Hardware solution**
  - ULINKplus Debug Adapter
    - μVision IDE
  - J-Trace Debug Probe
    - SystemView analyzer

- **Timing behavior**
  - RapiTask
  - RapiTime

- **OS behavior**
  - LTTng
  - Tracealyzer

**HPC**

- **Compile-time instrumentation**
  - Score-P
    - Scalasca
    - Vampire
    - TAU

- **Compile- and run-time instrumentation**
  - Extrae¹
    - Paraver

¹ https://tools.bsc.es/extrae
Analysis tools: from EC to HPC systems

**Hardware solution**
- ULINKplus Debug Adapter
  - μVision IDE
- J-Trace Debug Probe
  - SystemView analyzer

**Timing behavior**
- RapiTask
- RapiTime

**OS behavior**
- LTTng
- Tracealyzer

**EC**

**HPC**
- Score-P
  - Scalasca
  - Vampire
  - TAU

- Extrae
  - Paraver

**Compile-time instrumentation**

**Compile- and run-time instrumentation**

✅ Sampling
✅ Tracing
✅ Instrumentation
✅ Profiling

✅ Parallel model characterization
❌ Non-functional requirements

1 https://tools.bsc.es/extrae
Proposal: adapting Extrae to EC systems

Adapt to a embedded system

1. Static environment
2. RTOS
3. Specific architecture modules

Analyze NFR

1. Temperature and power consumption
2. Memory consumption
3. Tasks communication
Outline

- The characterization of OpenMP
- Accommodating Extrae to embedded systems: the GR740
- New functionalities in Extrae
- Analysis: correlating parallelism and non-functional requirements
- Conclusions
The characterization of OpenMP

- Thread-based model
- Task-based model

**Parallel Programming Model**
- Exposed parallelism
- Load balance
- Synchronization overhead
- Contention overhead
- Performance
- Power consumption
- Temperature

**Non-functional requirements**

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel loop chunk</td>
<td>Implicit task</td>
</tr>
<tr>
<td>Initial task</td>
<td>Explicit task</td>
</tr>
<tr>
<td>Master thread</td>
<td>Worker thread</td>
</tr>
</tbody>
</table>
**Embedded Systems: the GR740**

Radiation-hard SoC designed as the ESA Next Generation Microprocessor.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>- LEON4 SPARC V8 @250MHz</td>
<td>- RTEMS RTOS</td>
</tr>
<tr>
<td>- IEEE-754 floating point unit</td>
<td>- RCC cross compilation system</td>
</tr>
<tr>
<td>- 16KB instruction and data caches</td>
<td>- RTEMS-5.0 C/C++ real-time kernel</td>
</tr>
<tr>
<td>- 2MB write-back L2 cache</td>
<td>with support for SMP</td>
</tr>
<tr>
<td>- LEON4 Statistics Unit, L4stat</td>
<td>- Newlib</td>
</tr>
<tr>
<td>- AHB Bus</td>
<td>- L4stat driver</td>
</tr>
<tr>
<td>- Temperature sensor controller</td>
<td></td>
</tr>
<tr>
<td>- Timer units</td>
<td></td>
</tr>
</tbody>
</table>
Adapting Extrae to the GR740

1. Intercepting calls in a static environment
2. POSIX dependence
3. Retrieving function names
4. Trace generation
5. Supporting hardware counters
6. Statically defining the environment
Adapting Extrae to the GR740

1. Intercepting calls in a static environment:

   OpenMP Call → Extrae → OpenMP runtime

   ◆ **Vanilla Extrae:** LD_PRELOAD mechanism at runtime.
   ◆ **Adapted Extrae:** Symbol wrapping at compile time, using linker flags.

```c
int i, j;
Wrap_GOMP_parallel()
```

```
application.c  extrae.a  libgomp.a
```

```c
Real_GOMP_parallel()
```
Adapting Extrae to the GR740

2. POSIX dependence:

◆ Extrae relies on standard functions and structures from POSIX.

◆ Unfortunately, not all C standard libraries implement all POSIX functions.

◆ Newlib does not implement the ucontext structure, used for implementing the sampling mechanism. In the adaptation it has been replaced by hardware timers.
Adapting Extrae to the GR740

3/4. Retrieving function names and trace generation:

- Originally, Extrae obtains the symbol names of the executable using the binutils libraries targeting the binary from the file system.

- The binary is not available inside the board file system, since it is loaded in RAM. In the adaptation, Extrae now specifies the binary path and the use of a remote file system (NFS).

- This remote file system is also required for generating the final traces, where we also need to take into account the file system limitations (maximum file size, maximum size per write, etc)
Adapting Extrae to the GR740

5. Supporting hardware counters:

◆ Vanilla Extrae relies on PAPI library to gather the hardware counters of the system. PAPI does not support the GR740 architecture.

◆ The GR740 board provides the L4STAT unit, that implements hardware counters. This data is accessible through the L4STAT driver.

◆ We have extended Extrae to additionally support the L4STAT driver instead of just PAPI.
## Analysis: Applications & Aspects

<table>
<thead>
<tr>
<th>Applications</th>
<th>Evaluated aspects</th>
</tr>
</thead>
<tbody>
<tr>
<td>SparseLU loops</td>
<td>Memory: stack and heap</td>
</tr>
<tr>
<td></td>
<td>Temperature and power consumption</td>
</tr>
<tr>
<td>SparseLU tasks</td>
<td>Task communication</td>
</tr>
<tr>
<td>Image processing</td>
<td>Sampling</td>
</tr>
</tbody>
</table>
Analysis: SparseLU

SparseLU loops

```c
#pragma omp parallel private(kk)
for (...) // 3 iterations
#pragma omp single
    lu0(BENCH[kk*bots_arg_size+kk]);
#pragma omp for nowait schedule(dynamic)
for (...) 
    fwd(BENCH[kk*bots_arg_size+kk], BENCH[kk*bots_arg_size+jj]);
#pragma omp for schedule(dynamic)
for (...) 
    bdiv (BENCH[kk*bots_arg_size+kk], BENCH[ii*bots_arg_size+kk]);
```
Analysis: memory consumption

Runtime states

Stack
Analysis: memory consumption

Runtime states

Stack

The main thread uses more stack memory than the others.

Application uses stack size between 1000 and 3000
Analysis: memory consumption

Runtime states

Dynamic (de) allocation

Matrix allocation

Runtime allocations
Analysis: memory consumption

Runtime states

Dynamic (de) allocation

Heap

Heap does not decrement, since memory does not return to the OS although it is freed.
The temperature of the system is correlated with the CPU usage.
Analysis: power consumption

The power consumption can be calculated using the information about cpu usage.
Tasks dependencies can be represented inside the traces.
Analysis: sampling and the AMBA bus

Image processing

Sampling 10ms

Sampling 250ms

Parallel user functions
### Extrae extensions portability

<table>
<thead>
<tr>
<th>Extensions</th>
<th>Applicable to</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Temperature and power consumption</td>
<td>GR740 boards</td>
</tr>
<tr>
<td>2. Memory consumption</td>
<td>RTEMS operating systems</td>
</tr>
<tr>
<td>3. Tasks communication</td>
<td>OpenMP-compatible systems</td>
</tr>
</tbody>
</table>
Conclusions

● Currently embedded systems lack of tools to analyze applications performance at parallel programming level.

● HPC analysis tools do not support the analysis of non-functional requirements.

● Well-tested performance tools such as Extrae can be:
  ○ adapted to the constraints of embedded systems, e.g., RTEMS + GR740.
  ○ extended to analyze non-functional requirements, such as temperature and power consumption, a key aspect in embedded systems.
Experiences on the characterization of parallel applications in embedded systems with Extrae/Paraver

adrian.munera@bsc.es

Work partially funded from the HP4S (High-Performance Parallel Payload Processing for Space) project under ESA-ESTEC ITI contract Nº 4000124124/18/NL/CRS

ICPP2020