Parallelization and Power Reduction of Embedded Real-time Applications by OSCAR Compiler on ARM and Intel Multicores

Hironori Kasahara
Professor, Dept. of Computer Science & Engineering
Director, Advanced Multicore Processor Research Institute
Waseda University, Tokyo, Japan
IEEE Computer Society Multicore STC Chair
URL: http://www.kasahara.cs.waseda.ac.jp/
OSCAR Parallelizing Compiler

To improve effective performance, cost-performance and software productivity and reduce power

Multigrain Parallelization

coarse-grain parallelism among loops and subroutines, near fine grain parallelism among statements in addition to loop parallelism

Data Localization

Automatic data management for distributed shared memory, cache and local memory

Data Transfer Overlapping

Data transfer overlapping using Data Transfer Controllers (DMAs)

Power Reduction

Reduction of consumed power by compiler control DVFS and Power gating with hardware supports.
Earliest Executable Condition Analysis for Coarse Grain Tasks (Macro-tasks)

A Macro Flow Graph

A Macro Task Graph
PRIORITY DETERMINATION IN DYNAMIC CP METHOD

Critical path length: \(60 \times 0.80 + 100 \times 0.20 = 68\)
# Earliest Executable Conditions

**EEC: Control dependence + Data Dependence**

Control dependences show executions of MTs are decided

Data dependences show data accessed by MTs are ready

<table>
<thead>
<tr>
<th>Macrotask No.</th>
<th>Earliest Executable Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>(1) 3</td>
</tr>
<tr>
<td>3</td>
<td>2 4 OR (1) 3</td>
</tr>
<tr>
<td>4</td>
<td>4 5 AND [2 4 OR (1) 3]</td>
</tr>
<tr>
<td>5</td>
<td>3 OR (2) 4</td>
</tr>
<tr>
<td>6</td>
<td>5 OR (4) 6</td>
</tr>
<tr>
<td>7</td>
<td>(2) 4 OR (1) 3</td>
</tr>
<tr>
<td>8</td>
<td>(8) 9</td>
</tr>
<tr>
<td>9</td>
<td>(8) 10</td>
</tr>
<tr>
<td>10</td>
<td>8 9 OR 8 10</td>
</tr>
<tr>
<td>11</td>
<td>11 12 AND [9 OR (8) 10]</td>
</tr>
<tr>
<td>12</td>
<td>11 13 OR 11 12</td>
</tr>
<tr>
<td>13</td>
<td>(8) 9 OR (8) 10</td>
</tr>
<tr>
<td>14</td>
<td>2 15</td>
</tr>
</tbody>
</table>

- **MT2 may start execution after MT1 branches to MT2 and MT1 finish execution.**
- **MT3 may start execution after MT1 branches to MT3.**
- **MT6 may start execution after MT3 finish execution or MT2 branches to MT4.**
MTG of Su2cor-LOOPS-DO400

- Coarse grain parallelism \( \text{PARA\_ALD} = 4.3 \)
Data-Localization: Loop Aligned Decomposition

- Decompose multiple loop (Doall and Seq) into CARs and LRs considering inter-loop data dependence.
  - Most data in LR can be passed through LM.
  - LR: Localizable Region, CAR: Commonly Accessed Region

```plaintext
DO I=1,101
A(I)=2*I
ENDDO

DO I=1,100
B(I)=B(I-1)+A(I)+A(I+1)
ENDDO

DO I=2,100
C(I)=B(I)+B(I-1)
ENDDO
```
Multicore Program Development Using OSCAR API V2.0

Sequential Application Program in Fortran or C
(Consumer Electronics, Automobiles, Medical, Scientific computation, etc.)

OSCAR API for Homogeneous and/or Heterogeneous Multicores and manycores
Directives for thread generation, memory, data transfer using DMA, power managements

Parallelized API F or C program

Low Power Homogeneous Multicore Code Generation

- Proc0
  - Code with directives
  - Thread 0

Low Power Heterogeneous Multicore Code Generation

- Proc1
  - Code with directives
  - Thread 1

- Accelerator 1
  - Code

- Accelerator 2
  - Code

Server Code Generation

OpenMP Compiler

Generation of parallel machine codes using sequential compilers

Homegeneous Multicores from Vendor A (SMP servers)

Heterogeneous Multicores from Vendor B

Executible on various multicores

Accelerator Compiler/ User
Add “hint” directives before a loop or a function to specify it is executable by the accelerator with how many clocks

Waseda OSCAR Parallelizing Compiler
- Coarse grain task parallelization
- Data Localization
- DMAC data transfer
- Power reduction using DVFS, Clock/ Power gating

Hitachi, Renesas, NEC, Fujitsu, Toshiba, Denso, Olympus, Mitsubishi, Esol, Cats, Gaio, 3 univ.

OSCAR: Optimally Scheduled Advanced Multiprocessor API: Application Program Interface
Speedup with 2 cores for Engine Crankshaft Handwritten Program on Renesas RPX Multi-core Processor

Macrotask graph with a lot of conditional branches

Branches are fused to macrotasks for static scheduling

Macrotask graph after task fusion

Grain is too fine (us) for dynamic scheduling.

1.6 times Speed up by 2 cores against 1 core

1 core 2 core
OSCAR Compile Flow for Simulink Applications

1. **Generate MTG → Parallelism**
   - Simulink model

2. **Generate gantt chart → Scheduling in a multicore**
   - C code

3. **Generate parallelized C code using the OSCAR API → Multiplatform execution (Intel, ARM and SH etc)**

Generate C code using Embedded Coder

```c
#define VesselExtraction_step ( )

int thr1;
int thr2;
int thr3;

{  
  oscar_thread_create ( & thr1,  
    thread_function_001, (void*)1 );
  oscar_thread_create ( & thr2,  
    thread_function_002, (void*)2 );
  oscar_thread_create ( & thr3,  
    thread_function_003, (void*)3 );

  VesselExtraction_step_REAL ( );
  oscar_thread_join ( thr1 );
  oscar_thread_join ( thr2 );
  oscar_thread_join ( thr3 );
}
```
Speedups of MATLAB/Simulink Image Processing on Various 4core Multicores

(Intel Xeon, ARM Cortex A15 and Renesas SH4A)

Road Tracking, Image Compression: [http://www.mathworks.co.jp/jp/help/vision/examples](http://www.mathworks.co.jp/jp/help/vision/examples)
Parallel Processing on Simulink Model

- The parallelized C code can be embedded to Simulink using C mex API for HILS and SILS implementation.

Call sequential C code from the S-Function block

Sequential C Code

Call parallelized C code from the S-Function block

Parallelized C Code
OSCAR compiler gives us 11.55 times speedup for 16 cores against 1 core on SR16000 Power7 highend server.
Parallel Processing of JPEG XR Encoder on TILEPro64

Multimedia Applications: Sequential C Source Code

- AAC Encoder
- JPEG XR Encoder
- Optical Flow Calc.

OSCAR Compiler

Parallelized C Program with OSCAR API

API Analyzer + Sequential Compiler

Parallelized Executable Binary for TILEPro64

Speedup (JPEG XR Encoder)

- 55x speedup on 64 cores

- Default Cache Allocation
- Our Cache Allocation

(1) OSCAR Parallelization

(2) Cache Allocation Setting

Local cache optimization:
Parallel Data Structure (tile) on heap allocating to local cache
Performance of OSCAR Compiler on Intel Core i7 Notebook PC

- OSCAR Compiler accelerate Intel Compiler about 2.0 times on average

CPU: Intel Core i7 3720QM (Quad-core)
MEM: 32GB DDR3-SODIMM PC3-12800
OS: Ubuntu 12.04 LTS
110 Times Speedup against the Sequential Processing for GMS Earthquake Wave Propagation Simulation on Hitachi SR16000 (Power7 Based 128 Core Linux SMP)
9.6 Times Speedup on 12 cores Power 8 against the Sequential Processing for GMS Earthquake Wave Propagation Simulation

- IBM S812L
  - CPU: POWER8
    - 12 cores
    - Clock Frequency 3.026GHz
  - Memory: 60GB
  - OS: Redhat Linux 7.1
  - Backend Fortran compiler: IBM xlf 15.1.1
- Evaluation using medium size input data (12GB)
211 Times Speedup against the Sequential Processing using Sun Studio Compiler for GMS Earthquake Wave Propagation Simulation on Fujitsu M9000 Sparc 128 core SMP

- 100 times speedup on 128 cores against one core using OSCAR compiler
- 211 times speedup on 128 cores against original GMS program on one core using OSCAR Sun Studio Compiler
Low-Power Optimization with OSCAR API

Scheduled Result by OSCAR Compiler

VC0  VC1

MT1  MT2

MT3  MT4

Generate Code Image by OSCAR Compiler

void main_VC0() {

MT1

Sleep

MT3

MT4

void main_VC1() {

MT2

#pragma oscar fvcontrol ¥
((OSCAR_CPU(),0))

Sleep

MT2

#pragma oscar fvcontrol ¥
(1,(OSCAR_CPU(),100))

}
Power on 4 cores ARM CortexA9 with Android

http://www.youtube.com/channel/UCS43INYEIkC8i_KIgFZYQBQ

H.264 decoder & Optical Flow (Using 3 cores)

ODROID X2
Samsung Exynos4412 Prime, ARM Cortex-A9 Quad core
1.7GHz～0.2GHz, used by Samsung's Galaxy S3

<table>
<thead>
<tr>
<th></th>
<th>1 core</th>
<th>2 cores</th>
<th>3 cores</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>H.264</strong></td>
<td>1.07</td>
<td>-52.3%</td>
<td>-79.2%</td>
</tr>
<tr>
<td><strong>Optical flow</strong></td>
<td>0.72</td>
<td>0.36</td>
<td>0.30</td>
</tr>
</tbody>
</table>

- On the same 3 cores, the power control reduced the power to 1/5～1/7 against no power control.
- The power control reduced the power to 1/2～1/3 compared with the ordinary sequential execution on 1 core without power control.
Power Reduction on Intel Haswell 3cores

H.264 decoder & Optical Flow

H81M-A, Intel Core i7 4770k
Quad core, 3.5GHz ~ 0.8GHz

- The power consumption was reduced to 1/3 ~ 1/4 by OSCAR power control against no power control on the same 3 processor cores.
- The power reduced to 2/5 ~ 1/3 by the compiler power control against 1 core no power control.
Automatic Power Reduction by OSCAR Compiler on Intel Haswell 4 Core Multicore

- Power Consumption for real-time face detection was reduced to 2/5 -

Parallel processing of face detection program on Intel Haswell 4 cores

Average power consumption when automatic power reduction is applied

- Without power control:
  - 1 core: 27.61 W
  - 3 cores: 39.24 W

- With power control:
  - 1 core: 21.01 W (reduced to 3/5, -60.37%)
  - 3 cores: 15.55 W (reduced to 2/5, -43.68%)

Speedups against sequential execution at the fastest execution mode

- Without power control:
  - 1 core: 93.06 ms
  - 2 cores: 48.80 ms
  - 3 cores: 21.01 ms

- With power control:
  - 1 core: 27.61 ms
  - 2 cores: 18.41 ms (2.44 times speedup)
  - 3 cores: 15.55 ms

Parallelization flow of OpenCV face detection program

Input

Camera

Loop for searching changing sizes

Searching loop along x and y directions

Face detection processing

Drawing

Display

Power measurement on Intel Haswell board

- CPU: Intel Core i7 4770K
- Number of cores: 4
- Clock frequency: 3.5GHz~0.8GHz
- Mother board: ASUS H81M-A

Inserting power measurement circuit between PMIC and CPU
**OSCAR Vector Multicore and Compiler for Embedded to Servers with OSCAR Technology**

**Target:**
- Solar Powered with compiler power reduction.
- Fully automatic parallelization and vectorization including local memory management and data transfer.
Summary

- OSCAR Automatic Parallelizing and Power Reducing Compiler has succeeded speedup and/or power reduction of scientific applications including, medical applications including “Cancer Treatment Using Carbon Ion”, and “Drinkable Inner Camera”, industry application including “Automobile Engine Control”, and “Smartphone”, on various multicores from different vendors including Intel, ARM, Renesas and Fujitsu.

- In automatic parallelization, 110 times speedup for “Earthquake Wave Propagation Simulation” on 128 cores of IBM Power 7 against 1 core, 1.60 times for handwritten “ Automobile Engine Control” on Renesas 2 cores using SH4A, 55 times for “JPEG-XR Encoding for Capsule Inner Cameras” on Tilera 64 cores Tile64 manycore, 3.56, 3.12 and 3.45 times for “MATLAB/Simulink Vessel Detection program ” on 4 cores Intel Xeon, ARM Cortex A15 and Renesas SH4A respectively.

- In automatic power reduction, consumed powers for H.264 and optical flow were reduced to 1/2 or 1/3 using 3 cores of ARM Cortex A9 and Intel Haswell against ordinary single core execution and real-time Human face detection to 3/5 using 3 cores of Haswell.