Many-Core Processing for the LOFAR Software Telescope

Rob van Nieuwpoort
Astron, the Netherlands Institute for Radio Astronomy
nieuwpoort@astron.nl

Why Radio?

Credit: NASA/IPAC

Centaurus A, visible light and radio

Software radio telescopes (1)
- We cannot keep on building larger dishes
- Replace dishes with thousands of small antennas
- Combine signals in software

Software radio telescopes (2)
- Software telescopes are being built now
  - LOFAR: LOw Frequency Array (Netherlands, Europe)
  - ASKAP: Australian Square Kilometre Array Pathfinder
  - MeerKAT: Karoo Array Telescope (South Africa)

- Future: SKA, Square Kilometre Array
  - Exa-scale! (10^{18}: giga, tera, peta, exa)

LOFAR overview
- Hierarchical
- Receiver
- Tile
- Station
- Telescope

- Central processing
  - Groningen
  - IBM BG/P
  - Dedicated fibers
### LOFAR low-band antennas

- 88,000 antennas
- 10–250 MHz
- 100x more sensitive
- Omnidirectional
- Hundreds of gigabits/s: 14x LHC
- Tens – hundreds of teraFLOPS

### LOFAR high-band antennas

### Station (150m)

### Characteristics

- 88,000 antennas
- 10–250 MHz
- 100x more sensitive
- Omnidirectional
- Hundreds of gigabits/s: 14x LHC
- Tens – hundreds of teraFLOPS

### LOFAR science

- Imaging
- Epoch of re-ionization
- Cosmic rays
- Extragalactic surveys
- Transients
- Pulsars
A LOFAR observation

- Cas A
- Supernova remnant
- 115–160 MHz
- 12 stations

Processing overview

BG/P performance

Correlator is $O(n^2)$ achieve 96% of the theoretical peak

Many-cores

- Intel core i7 quad core + hyperthreading + SSE
- Sony/Toshiba/IBM Cell/B.E. QS21 blade
- GPUs: NVIDIA Tesla C1060/GTX280, ATI 4870
- Compare with production code on BG/P
- Compare architectures
  - Implemented everything in assembly
  - [ICS 2009], [IJPP 2010], [IEEE Signal processing 2010]

Essential many-core properties

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Real core MHz</th>
<th>MM BG/P</th>
<th>ATI 4870</th>
<th>NVIDIA C1060</th>
<th>STI Cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPU s</td>
<td>16</td>
<td>3</td>
<td>880</td>
<td>240</td>
<td>32</td>
</tr>
<tr>
<td>GFlops</td>
<td>85</td>
<td>14</td>
<td>1200</td>
<td>936</td>
<td>205</td>
</tr>
<tr>
<td>loat registers</td>
<td>64</td>
<td>64</td>
<td>4096</td>
<td>2048</td>
<td>512</td>
</tr>
<tr>
<td>Device RAM bandwidth (GB/s)</td>
<td>n.a</td>
<td>n.a.</td>
<td>115.2</td>
<td>102</td>
<td>n.a.</td>
</tr>
<tr>
<td>host RAM bandwidth (GB/s)</td>
<td>25.6</td>
<td>13.6</td>
<td>8.5</td>
<td>9.0</td>
<td>9.6</td>
</tr>
<tr>
<td>per operation bandwidth slowdown compared to BG/P</td>
<td>3.3</td>
<td>1.0</td>
<td>10.4</td>
<td>9.2</td>
<td>7.9</td>
</tr>
</tbody>
</table>

Correlator algorithm

- For all channels (63488)
- For all combinations of two stations (2080)
- Complex float sum = 0;
- For the time integration interval (768 samples)
  - Sum += sample1 * ~sample2 (complex multiplication)
- Store sum in memory
- Straightforward implementation has 1 flop / byte!
Correlator optimization

- Overlap data transfers and computations
- Exploit caches / shared memory / local store
- Loop unrolling
- Tiling (needs registers)
- Scheduling
- SIMD operations
- ...

Correlator performance

- Measured power efficiency
  - Current CPUs (even at 45 nm) still are less power efficient than BG/P (90 nm)
  - GPUs are not 15, but only 2-3x more power efficient than BG/P
  - 65 nm Cell is 4x more power efficient than the BG/P

Weak and strong points

- Intel Core i7
  - well-known toolchain
  - L2 prefetch unit
  - high memory bandwidth
- IBM BG/P
  - largest # cores
  - shuffling support
- ATI 4870
  - CUDA is high-level
  - explicit cache (LS)
  - shuffle capabilities
- NVIDIA Tesla C1060
  - low PCI-e bandwidth (4.6 GB/s)
  - transfer slows down kernel
  - CAL is low-level
  - bad Brook++ performance
- STI Cell
  - low PCI-e bandwidth (5.6 GB/s)
  - no increment in odd pipeline

Conclusions

- Software telescopes are the future, extremely challenging
- Software provides the required flexibility
- Many-core architectures show great potential
- PCI-e is a bottleneck
- Compared to the BG/P or CPUs, the many-cores have low memory bandwidth per operation
- This is OK if the architecture allows efficient data reuse
  - Optimal use of registers (tile size + SIMD strategy)
  - Exploit caches / local memories / shared memories
  - The Cell has 8 times lower memory bandwidth per operation, but still works thanks to explicit cache control and large number of registers

Questions?

Rob van Nieuwpoort
nieuwpoort@astron.nl