The best of both worlds:
Delivering aggregated performance for high-performance math libraries in accelerated systems

Dr James Irwin and Mr Simon McIntosh-Smith
ClearSpeed Technology plc
Many systems are already reaching infrastructure limits:

- Data center size
- Power supply
- Cooling

**Accelerators emerging to significantly increase performance per (cubic meter, watt)**

Tokyo Tech created the first of the new wave of accelerated supercomputers, TSUBAME

- Performance increased from 38 TFLOPS to 47 TFLOPS with 360 ClearSpeed Advance™ accelerators
- An increase in performance of 24%, but for just a 1% increase in power consumption
- #9 in the November 2006 Top500

Professor Matsuoka standing beside TSUBAME at Tokyo Tech
ClearSpeed accelerators and CSXL

ClearSpeed Advance™ accelerator:
- ~66 GFLOPS DGEMM @210MHz
- PCI-X now, PCI Express x8 soon
- ~25 watts for entire board
- 1GByte of local DRAM with ECC
- 20cm long

- CSXL accelerated math library includes key routines from L3 BLAS and LAPACK
  - E.g. DGEMM, DGETRF

- Plug-and-play: applications call routines in CSXL like any other BLAS/LAPACK library
  - Data transfers to/from accelerator handled internally
ClearSpeed’s visual profiler showing the LINPACK benchmark running on an unaccelerated 1.8GHz dual-core AMD Opteron
Accelerators: replacement performance

- Typically an accelerator runs a computationally intensive kernel instead of the host.
- This leaves performance on the table:
  - Today’s multi-core CPUs are capable of tens of GFLOPS.
Accelerators: additive performance

Extend CSXL so that it can heterogeneously use the host and the accelerator at the same time.
Additive LINPACK

DGEMM now using the dual core Opteron and the accelerator at the same time

• DGEMM performance increased from 46 to 60 GFLOPS
Static load balancing

• Once CSXL can use the host and the accelerator to perform DGEMM at the same time, how should it divide the work from a single DGEMM call across these heterogeneous resources?

• A simple approach is to use a *static host fraction*
  – Split every call to DGEMM into a fixed amount for the host and the rest for the accelerator

• **Advantages of this approach:**
  – Can be very finely tuned by experimentation
  – Simple to implement

• **Disadvantages of this approach:**
  – Inflexible – different sizes and shapes of DGEMM may suit different host fractions
  – Time consuming – deriving the ideal host fraction may require much experimentation
    • Static host fraction derived as the ratio of the DGEMM performance curves of the host and the accelerator
Dynamic host fraction and auto calibration

• The static host fraction scheme has drawbacks

• We developed a dynamic host fraction method which employs a model of the performance of a homogeneous system when multiplying an $m \times k$ matrix by a $k \times n$ matrix:

\[
d_1 mn + d_2 mk + d_3 nk + d_4 mnk
\]

• The values of the $d_i$ coefficients can be derived automatically as part of a calibration step

• Advantages over the static scheme:
  – Flexible: works for varying shapes and sizes of DGEMM
  – Easy to use: auto calibration
Test system specifications

• We tested the effectiveness of the dynamic host fraction scheme in two different multi-core systems:

1. A four core system
   – Two AMD Opteron 280s (2.4GHz dual-core)

2. An eight core system
   – Four AMD Opteron 870s (2GHz dual-core)

• Both systems included a single ClearSpeed accelerator
Predicted vs. measured DGEMM performance

- Individually validating the models of the hosts and the accelerators

4 core system

8 core system
Dynamic host fractions

- Resulting dynamic host fractions

4 core system 8 core system
Measured aggregate DGEMM performance

- Do we achieve additive performance?

4 core system

8 core system
How good are these results?

- **What additive performance might we expect for DGEMM?**
  - One simple model would be to sum the performance predicted by the models for the host and the accelerator.
  - Ignores important effects:
    - Overhead on the host supporting the accelerator
    - Peaky DGEMM performance
Additive LINPACK results

Static (took a morning to manually calibrate):
- 4 core system: 41.3 GFLOPS
- 8 core system: 43.6 GFLOPS

Dynamic (auto calibrated in 10 seconds):
- 4 core system: 35.79 GFLOPS
  - 87% of best static score
- 8 core system: 39.57 GFLOPS
  - 91% of best static score

- LINPACK job size was ~6GBytes, relatively small
Conclusions

• We have demonstrated that aggregating heterogeneous compute resources across multi-core general purpose CPUs and accelerators can be efficient and flexible
  – We believe this method is widely applicable within HPC

• The heterogeneous DGEMM implementation in both static and dynamic host fraction forms achieved excellent aggregate performance

• The dynamic host fraction version was significantly easier to use, and delivers better performance in situations where the shapes of the DGEMM calls varied
Further Work

• More accurate models and calibration of \textit{DGEMM} performance on the host and accelerator

• Extend heterogeneous support across:
  – Other BLAS/LAPACK functions, such as \textit{DTRSM}
  – Other data parallel computations

• Support multiple accelerators simultaneously

• Extend the range of supported heterogeneous processor classes to include remote resources

• Optimize for performance per watt vs. outright performance
Acknowledgements

• We would like to thank our colleagues at ClearSpeed who have been developing the CSXL library and who have significantly contributed to this work, including:
  – Matthias Dejaegher
  – John Gustafson

• Our thanks go to the entire CSXL team!

• A full of this paper will be available online from www.clearspeed.com