The Evolution of Programmable Architectures for Accelerating Science

Ian Buck
VP/GM Accelerated Computing
SMC 2014
Sept 4, 2014
translating transistors into performance

- 1.8x increase of transistors
- 20% decrease in clock rate
- 6.6x GFLOP speedup
C = A + B
A simple, explicit programming language solution

Extend only where necessary

Simple thread id with function

```cpp
__global__
void add_parallel(int n,
  float *c,
  float *a,
  float *b)
{
  int i = blockIdx.x*blockDim.x + threadIdx.x;
  if (i < n) c[i] = a[i] + b[i];
}

// Perform c=a+b on 1M elements
add_parallel<<<4096,256>>>(n, c, a, b);
```
Removing divergence pain from parallel programming

- **SIMD**: Single Instruction Multiple Data
- **MIMD**: Multiple Instruction Multiple Data
- **SIMT**: Single Instruction Multiple Thread

**SIMT = MIMD Programming Model w/ SIMD Implementation Efficiencies**
SIMT: Eliminates fully predicated vectors

Managing divergence becomes an optimization rather than a requirement.

Slide courtesy Andrew Glew, “Coherent Threading” Berkeley ParLab Preso
Directives & Implicit Parallelism

Program myscience
... serial code ...
!$acc kernels
do k = 1,n1
do i = 1,n2
  ... parallel code ...
endo
dendo
!$acc end kernels
...
End Program myscience

Insert compiler hints

Compiler auto-parallelize code with limited effort

Architecture assisted runtimes aid in managing the data
NVLink: Data Transfer At Speed of CPU Memory

Stacked Memory

HBM 1 Terabyte/s

Pascal

interposer

Package Substrate

DDR Memory

DDR4 50-75 GB/s

NVLink 80 GB/s

Directive runtimes get baseline performant access to bulk memory
Rapid Parallel C++ Development

- Resembles C++ STL
- Open source
- High-level interface
  - Enhances developer productivity
  - Enables performance portability between GPUs and multicore CPUs
- Flexible
  - CUDA, OpenMP, and TBB backends
  - Extensible and customizable
  - Integrates with existing software

```cpp
// generate 32M random numbers on host
thrust::host_vector<int> h_vec(32 << 20);
thrust::generate(h_vec.begin(), h_vec.end(), rand);

// transfer data to device (GPU)
thrust::device_vector<int> d_vec = h_vec;

// sort data on device
thrust::sort(d_vec.begin(), d_vec.end());

// transfer data back to host
thrust::copy(d_vec.begin(), d_vec.end(), h_vec.begin());
```

Inline Parallelism: C++

Language features enable parallelism in-line with sequential code

See RAJA (LLNL) and Kokkos (Sandia)

```cpp
std::for_each( std::par, std::begin(options), std::end(options), [](Option &i) {
    const double d1 = (log((i.S/i.X))+(i.r+i.v*i.v/2)*i.T)/(i.v*sqrtf(i.T));
    const double d2 = d1-i.v*sqrtf(i.T);
    i.call = i.S * CND(d1)-i.X * exp(-i.r*i.T)*CND(d2);
    i.put = i.X * exp(-i.r * i.T) * CND(-d2) - i.S * CND(-d1);
});
```

Example: Parallel Black-Scholes kernel in (future) standard C++

- Standard parallel algorithms library (Projected C++17)
- Lambda, std::begin/end (C++11)
- Can substitute vendor-specific execution policy (std::par → nvidia::gpu)
Retarget Inline Parallelism at Run Time

- Compile application once
- Runtime Compilation of `for_each/lambda`
  - Based on GPU availability, problem size, etc.
  - Data-dependent code specialization, fusion, etc.

```cpp
std::for_each(
    Lambda
);  
```
Diversity of Programming Languages

ActionScript  Ada  Assembly  Autoconf  Automake  AWK  BlitzMax  Boo  Brainfuck  Brainfuck++  C  C#

C++  C/C++  ChaiScript  Classic Basic  ClearSilver  Clojure  CMake  CoffeeScript  CSS  CUDA  D  DCL  DOS

batch  script  Dylan  Ebuild  eC  Eiffel  Emacs  Lisp  Erlang  Exheres  F#  Factor  forth  Fortran  (Fixed-format)  Fortran

(Free-format)  Go  Groovy  Haml  Haskell  HaXe  HTML  IDL/PV-WAVE/GDL  Jam  Java

JavaScript  Limbo  Lisp  Logtalk  Lua  Make  Matlab  MetaFont  MetaPost  Modula-2  Modula-3  MXML  Nix

NSIS  Oberon  Objective-C  Objective Caml  Objective-J  Octave  OpenGL  Shading  Pascal  Perl  PHP  Pike

Prolog  Puppet  Python  QML  R  Racket  REBOL  Rexx  Ruby  Scala  Scheme  Scilab  shell script  Smalltalk  SQL

Stratego  Structured Basic  Tcl  TeX/LaTeX  Vala  VHDL  Vim Script  Visual Basic  XAML  XML  XML Schema

XSL  Transformation

http://www.ohloh.net
A long term viable architecture for science is a leveraged architecture
Machine Learning

“Okay, Google…”

Hi. I’m Cortana.
Baidu

Google of China
#5 Internet Site

Every Day:
5B+ queries
500M+ users
100M+ mobile users
100M+ photos

<table>
<thead>
<tr>
<th>Storage</th>
<th>&gt;2000PB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing</td>
<td>10-100PB/day</td>
</tr>
<tr>
<td>Webpages</td>
<td>100b-1000b</td>
</tr>
<tr>
<td>Index</td>
<td>100b-1000b</td>
</tr>
<tr>
<td>Update</td>
<td>1b-10b/day</td>
</tr>
<tr>
<td>Log</td>
<td>100TB~1PB/day</td>
</tr>
</tbody>
</table>

Baidu Data Sets and Training

- Image recognition: 100 millions
- OCR: 100 millions
- Speech: 10 billions

Training data projected to grow 10X per year

Training time: Weeks to months on clusters of GPUs
Deep Neural Nets:

Image

SGEMM()
The Day Job That Makes It All Possible…

- Leverage volume graphics and service provider market to serve scientific computing

Mobile

Gaming

Pro Vis