GPUs

Johannes Coym

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

2016-12-08

Gliederung (Agenda)

1 Einleitung

2 Technologien

3 Beispiele

4 Ausblick

5 Literatur

Einleitung

- Vorteile gegenüber CPUs
  - Extrem hohe Anzahl von Rechenkernen
  - Optimiert für parallelisierbare Berechnungen
  - Können die parallelisierbaren Aufgaben übernehmen um die CPU zu entlasten

Einleitung

- Nachteile gegenüber CPUs
  - Lediglich für spezielle Berechnungen ausgelegt
  - Für den normalen Gebrauch ungeeignet
  - Programme müssen komplett umgeschrieben werden für Berechnungen auf der GPU

Einleitung

- Leistung gegenüber CPUs
  - Intel Core i7 6700k: ~0,2 TFLOPS
  - Intel Xeon E5 2699 v4: ~1,4 TFLOPS[1]
  - Nvidia Tesla P100: ~5,3 TFLOPS[2]

Grafikschnittstellen

- DirectX
  - Grafikschnitstelle von Microsoft
  - Bei Spielen für Windows meistens verwendet
- OpenGL/Vulkan
  - Offene Grafikschnittstelle
  - Betriebssystemübergreifend verfügbar

OpenCL

- Offene Schnittstelle zur Verwendung der GPU für nicht-grafische Anwendungen
- Ursprünglich von Apple entwickelt und seit 2008 standardisiert
- Enthält eigene Programmiersprachen “OpenCL C“ und “OpenCL C++“
  - Basierend auf C99, bzw. C++14
  - Durch einige neue Datentypen erweitert, jedoch manche Funktionen von C(++) nicht verfügbar
- Enthält ebenfalls die Möglichkeit unter Windows direkt auf DirectX- oder OpenGL-Objekte zuzugreifen[3]

OpenCL

- Unterstützt von GPUs von:
  - AMD Radeon HD5450 und neuer(2010)
  - ARM Mali 604 GPU und neuer(2010)
  - Intel Core Prozessoren ab der 3. Generation(2012)
  - Nvidia Geforce 8000 Reihe und neuer(2007)

OpenACC

- Offene Schnittstelle für die parallele Verwendung von CPU und GPU im Programm
- Durch Markieren bestimmter Codeteile, werden diese auf der GPU berechnet
- Der Quellcode muss nicht neu geschrieben werden um die GPU Leistung zu verwenden, der Compiler übernimmt diesen Teil
- Für die Programmiersprachen C, C++ und Fortran
- Kann auch zusammen mit Technologien wie OpenMP für Projekte verwendet werden

OpenACC

- Vier verscheidene Klimaberechnungen
  - Auf einer 8 Kern Intel Sandy Bridge CPU unter Verwendung von MPI: 57,6s
  - Auf einem CPU Kern und einer Nvidia K20X GPU unter Verwendung von OpenACC: 13,7s [4]

CUDA

- Schnittstelle von Nvidia zur Verwendung von Nvidia GPUs für nicht-grafische Anwendungen
- Beinhaltet Implementationen von OpenCL und OpenACC
- Findet Verwendung in hunderten Anwendungen in Bereichen, wie Datenwissenschaft, Maschinelles Lernen, Physik und Klimaberechnungen[5]

CUDA

- Bietet 4 Optionen zur Beschleunigung von Anwendungen mit Hilfe der GPU[6]:
  - Bibliotheken zur Beschleunigung bestimmer Prozesse
  - OpenACC zur Beschleunigung bestimmter Codeabschnitte
  - Eigene CUDA C/C++ Programmiersprachen für parallele Programmierung
  - Ein eigenes SDK für Deep Learning

OpenCL C

```
1  __kernel void SAXPY (__global float* x, __global
       ↪ float* y, float a)
2  {
3  const int i = get_global_id (0);
4  y [i] += a * x [i];
5  }
6  ...
7  size_t global_size = VECTOR_SIZE;
8     size_t local_size = 64;
9     clStatus = clEnqueueNDRangeKernel(command_queue,
          ↪ kernel, 1, NULL, &global_size, &local_size,
          ↪ 0, NULL, NULL);
```

Listing 1: SAXPY OpenCL C[8]

CUDA C++

```
1  __global__ void saxpy(int n, float a, float *
       ↪ restrict x, float * restrict y)
2  {
3      int i = blockIdx.x*blockDim.x + threadIdx.x;
4      if (i < n) y[i] = a*x[i] + y[i];
5  }
6  ...
7  int N = 1<<20;
8  cudaMemcpy(d_x, x, N, cudaMemcpyHostToDevice);
9  cudaMemcpy(d_y, y, N, cudaMemcpyHostToDevice);
10
11 // Perform SAXPY on 1M elements
12 saxpy<<<4096,256>>>(N, 2.0, x, y);
13
14 cudaMemcpy(y, d_y, N, cudaMemcpyDeviceToHost);
```

Listing 2: SAXPY CUDA C[9]

OpenACC

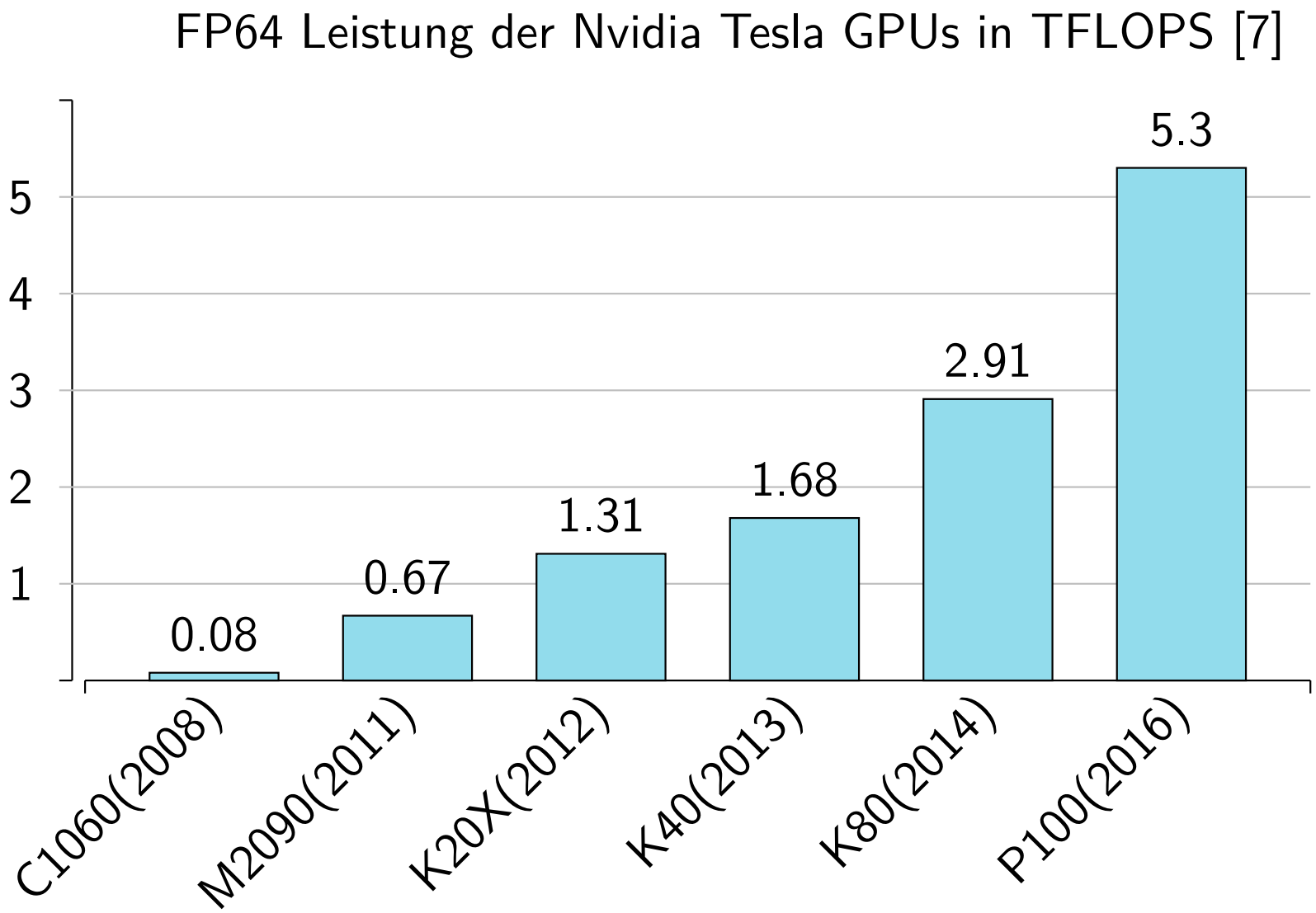
```
1 void saxpy(int n, float a, float * restrict x, float
      ↪ * restrict y)
2 {
3 #pragma acc kernels
4     for (int i = 0; i < n; ++i)
5         y[i] = a*x[i] + y[i];
6 }
7 ...
8 // Perform SAXPY on 1M elements
9 saxpy(1<<20, 2.0, x, y);
```

Listing 3: SAXPY OpenACC[9]

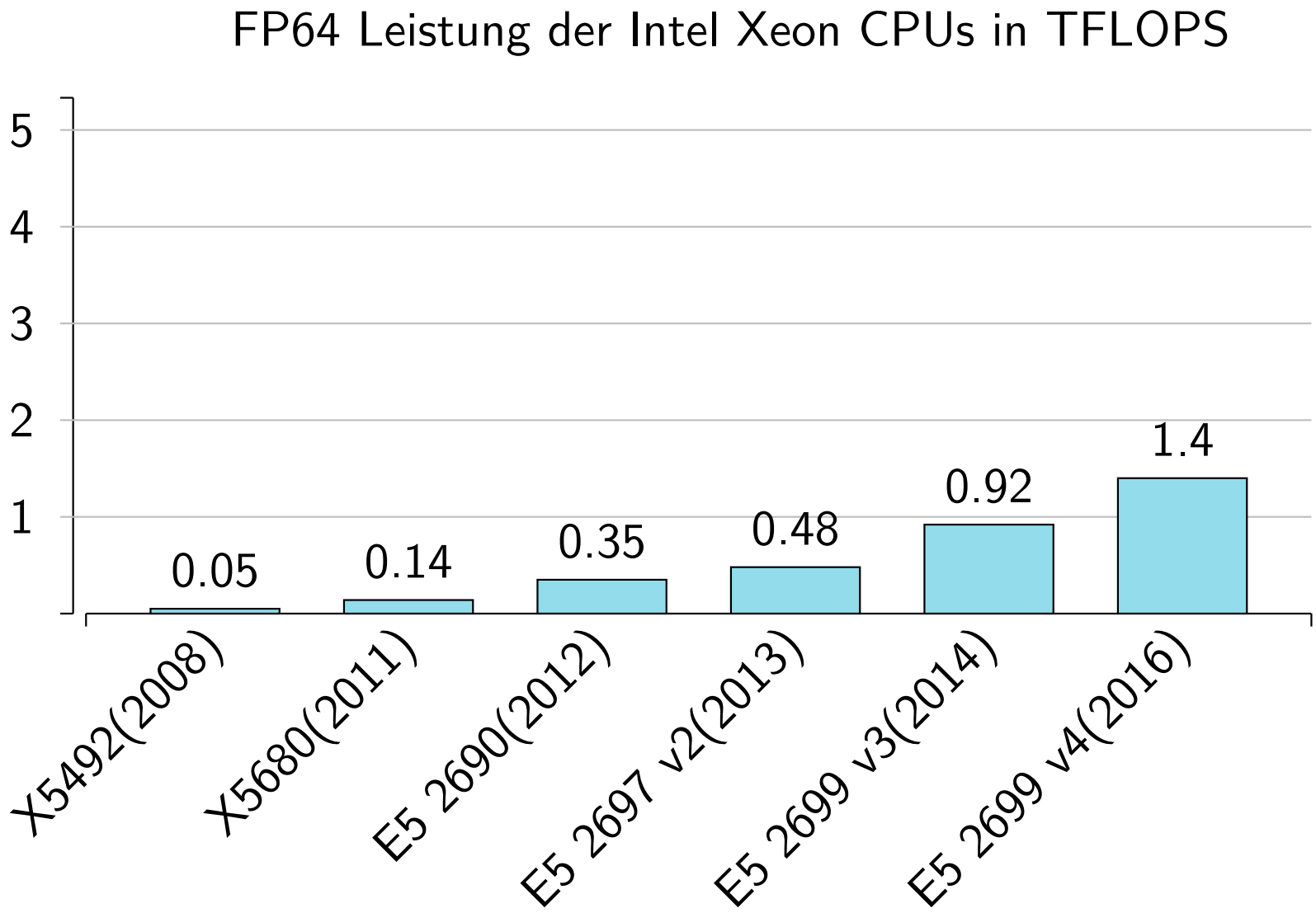
OpenACC

- Quellen:
  - https://devblogs.nvidia.com/parallelforall/openacc-example-part-1/, Zugriff 30.11.2016
  - https://github.com/parallel-forall/code-samples/tree/master/posts/002-openacc-example, Zugriff 30.11.2016

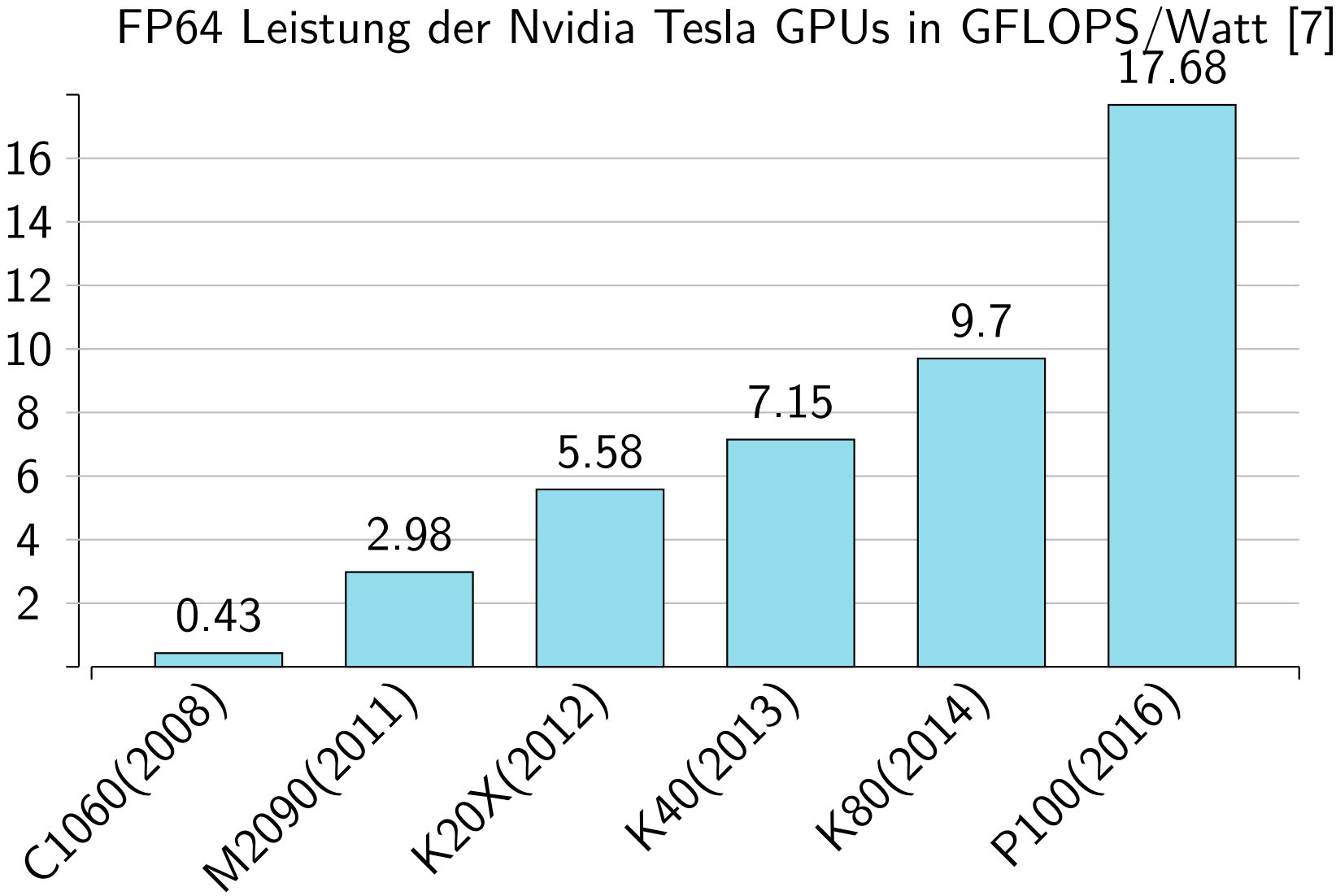
Entwicklung GPU Leistung



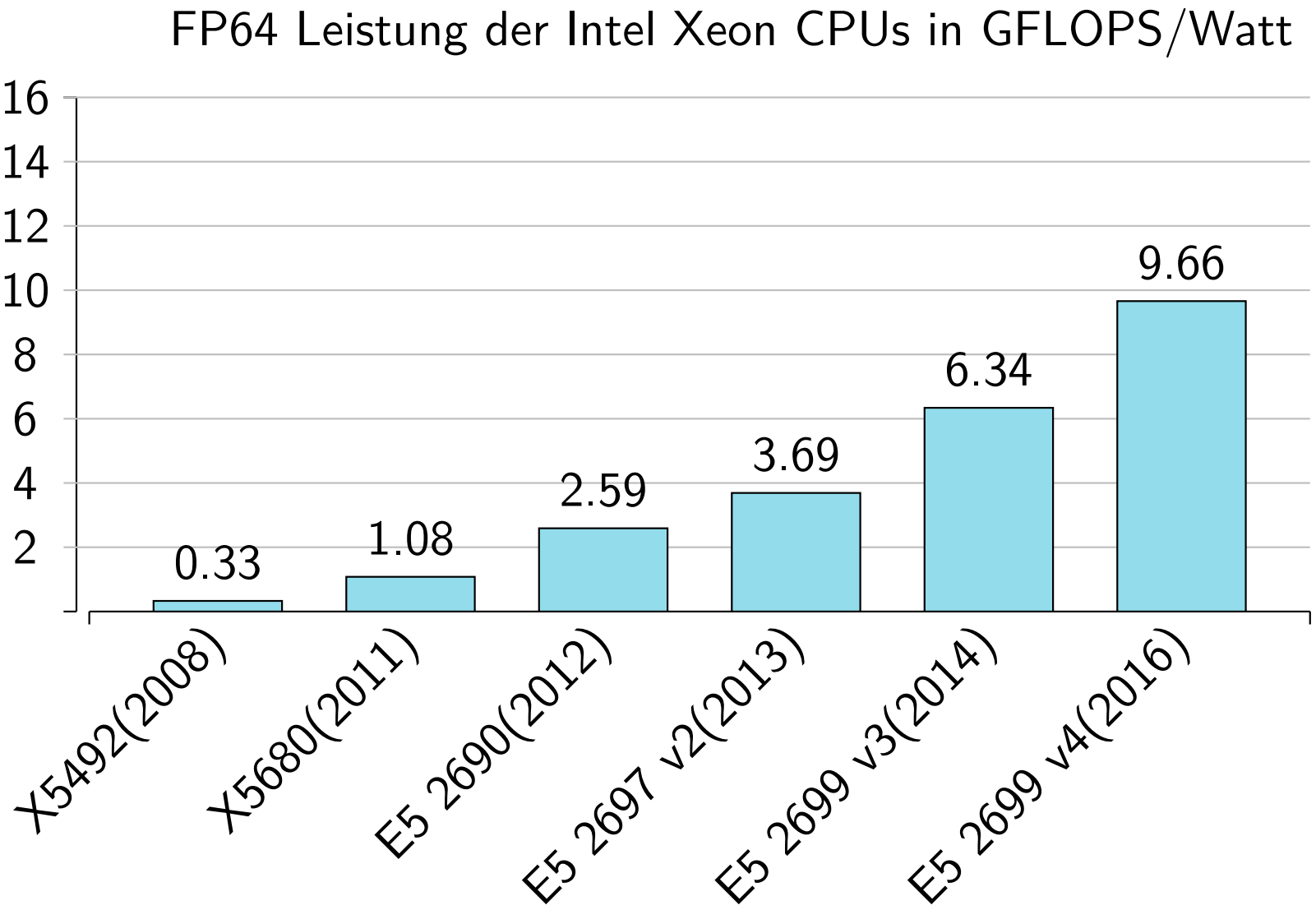
Entwicklung CPU Leistung



Entwicklung GPU Leistung und Stromverbrauch



Entwicklung CPU Leistung und Stromverbrauch



Ausblick

- Wachsender Leistungsunterschied zwischen CPUs und GPUs für bestimmte Berechnungen
  - Dadurch zunehmende Bedeutung von GPUs in diesen Bereichen
- Durch diese Spezialisierung ist es jedoch eher nicht absehbar, dass irgendwann reine GPU-Systeme kommen
- Konkurrenz für GPUs kommend durch Many-Core-CPUs, wie die Xeon Phis, welche auch eigenständig laufen können

Zusammenfassung Technologien

- OpenCL
  - Hardwarenahe Benutzung für diverse Plattformen
  - Kompliziert zu benutzen
- CUDA
  - Hardwarenahe Benutzung spezialisiert und optimiert für Nvidia GPUs
  - Leichterer Einstieg als OpenCL
- OpenACC
  - Compiler-basierte Schnittstelle für diverse Plattformen
  - Leicht für einzelne Programmteile einzubinden

Literatur

[1] http://en.community.dell.com/techcenter/high-performance-computing/b/general_hpc/archive/2016/03/31/measuring-performance-of-intel-broadwell-processors-with-high-performance-computing-benchmarks, Zugriff 22.11.2016

[2] http://www.nvidia.de/object/tesla-p100-de.html, Zugriff 22.11.2016

[3] https://de.wikipedia.org/wiki/OpenCL, Zugriff 22.11.2016

[4] http://www.openacc.org/content/applications/Weather, Zugriff 24.11.2016

Literatur

[5] http://www.nvidia.de/object/gpu-computing-applications-de.html, Zugriff 29.11.2016

[6] https://developer.nvidia.com/accelerated-computing, Zugriff 30.11.2016

[7] https://en.wikipedia.org/wiki/Nvidia_Tesla, Zugriff 30.11.2016

[8] https://www.packtpub.com/mapt/book/Application-Development/9781849692342/1/ch01lvl1sec12/An+example+of+OpenCL+program, Zugriff 5.12.2016

[9] https://devblogs.nvidia.com/parallelforall/six-ways-saxpy/, Zugriff 5.12.2016