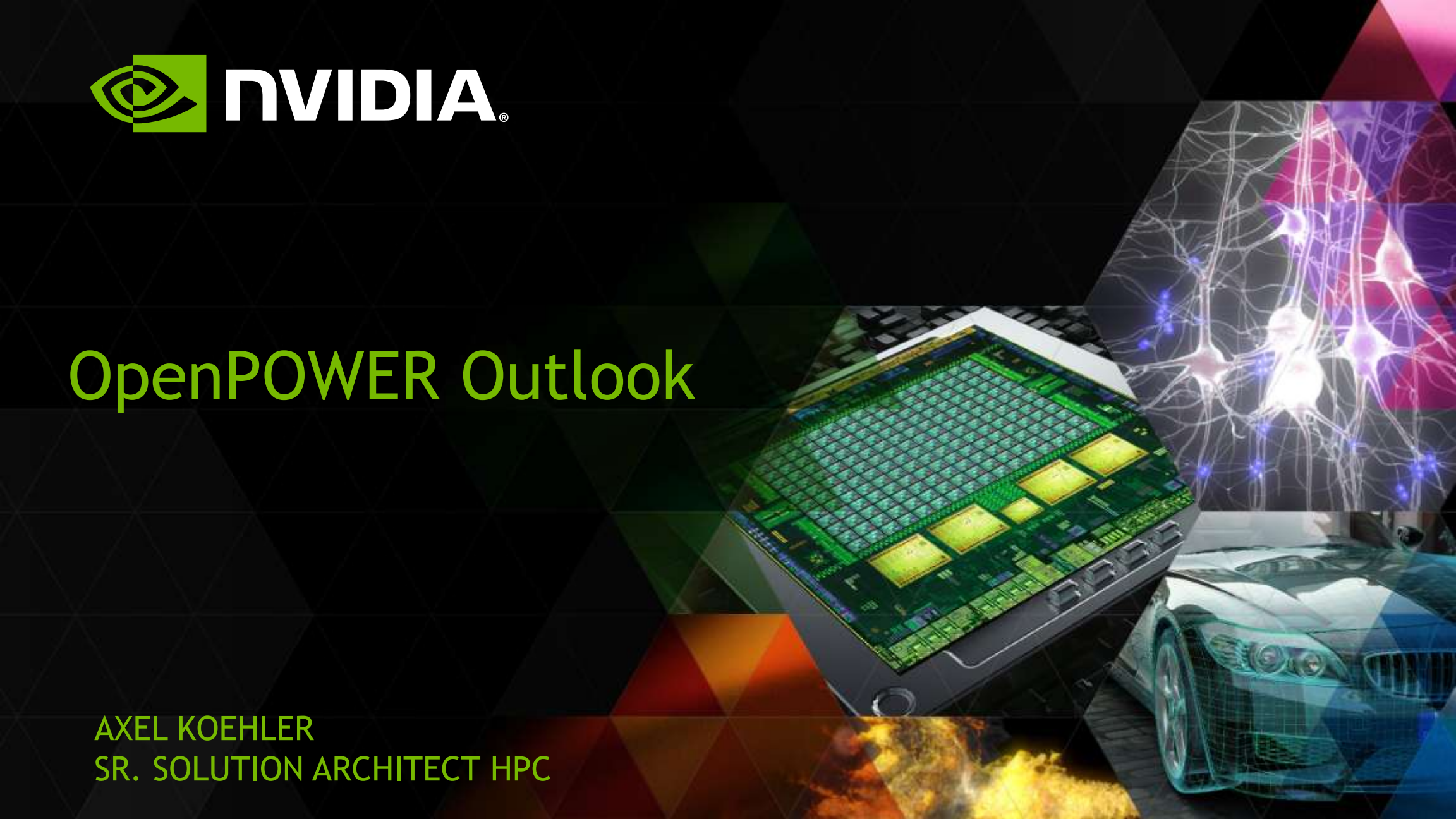


OpenPOWER Outlook

AXEL KOEHLER
SR. SOLUTION ARCHITECT HPC

Driving industry innovation

The goal of the OpenPOWER Foundation is to create an open ecosystem, using the POWER Architecture to share expertise, investment, and server-class intellectual property to serve the evolving needs of customers.

– Opening the architecture to give the industry the ability to innovate across the full Hardware and Software stack
  - Simplify system design with alternative architecture
  - Includes SOC design, Bus Specifications, Reference Designs, FW OS and Open Source Hypervisor
  - Little Endian Linux to ease the migration of software to POWER
– Driving an expansion of enterprise class Hardware and Software stack for the data center
– Building a complete ecosystem to provide customers with the flexibility to build servers best suited to the Power architecture

Giving ecosystem partners a license to innovate

OpenPOWER will enable data centers to rethink their approach to technology.

Member companies may use POWER for custom open servers and components for Linux based cloud data centers.

OpenPOWER ecosystem partners can optimize the interactions of server building blocks – microprocessors, networking, I/O & other components – to tune performance.

How will the OpenPOWER Foundation benefit clients?

- OpenPOWER technology creates greater choice for customers
- Open and collaborative development model on the Power platform will create more opportunity for innovation
- New innovators will broaden the capability and value of the Power platform

What does this mean to the industry?

- Game changer on the competitive landscape of the server industry
- Will enable and drive innovation in the industry
- Provide more choice in the industry

Platinum Members

Proposed Ecosystem Enablement

Software

Power Open Source Software Stack Components

Cloud Software — openstack

Standard Operating Environment (System Mgmt) — Ganglia, CAT, oVirt

Operating System / KVM — Linux

Existing Open Source Software Communities

Firmware — OpenPOWER Firmware

Hardware — OpenPOWER Technology

New OSS Community

System Operating Environment Software Stack

A modern development environment is emerging based on tools and services

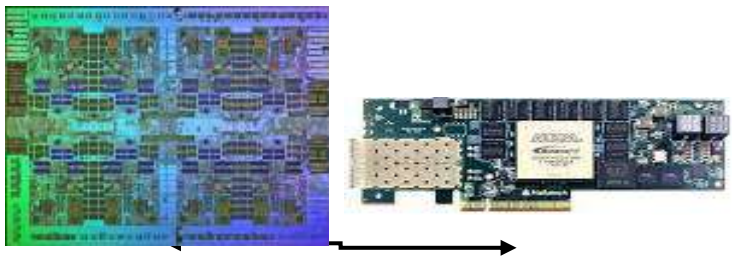
Bootstrap, node, SOCKET IO, redis, Cassandra, RabbitMQ, NGINX, mongoDB, MariaDB, HA PROXY

40,000 packages now available

Hardware

Multiple Options to Design with POWER Technology Within OpenPOWER

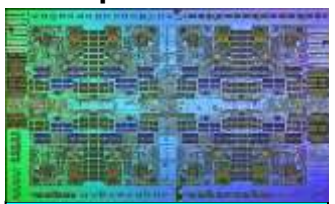
POWER8



CAPI over PCIe

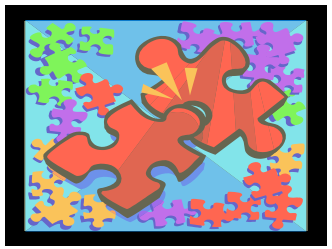
"Standard POWER Products" – 2014

Framework to Integrate System IP on Chip

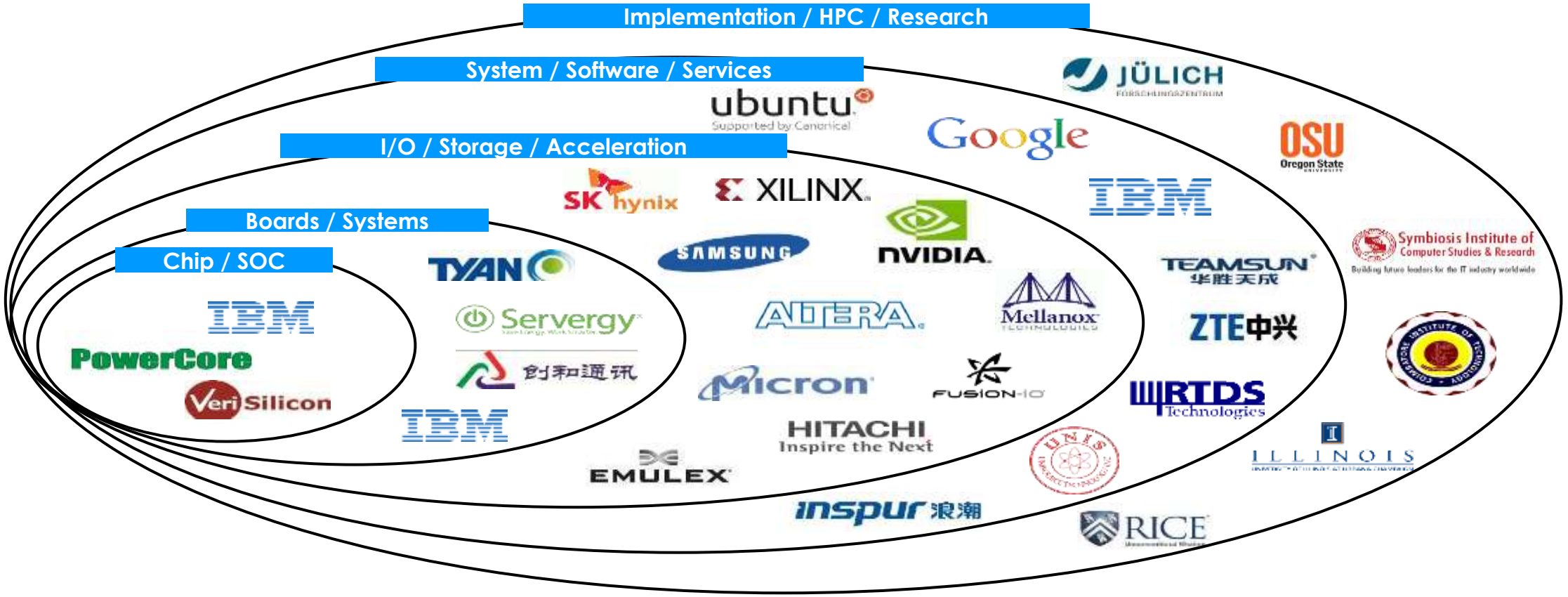


"Custom POWER SoC" – Future

Industry IP License Model



Building collaboration and innovation at all levels



Welcoming new members in all areas of the ecosystem

150+ inquiries and numerous active dialogues underway

Proposed Work Groups and Projects

Work Group	Projects	Participants
System Software (Open Source)	• Linux LE • KVM • Firmware –OpenPOWER FW interface • POWER LE ABI	Public Public Public Public
Application Software (Open Source)	• System Operating Environment – OpenPOWER Software ecosystem enablement • Toolchain	Public Public
Open Server Development Platform	• Power 8 Developer Board • POWER 8 Reference Design	Member Member
Hardware Architecture	• OpenPOWER profile of architecture – Power8 ISA Book 1, 2, 3 • Coherent Accelerator Interface Architecture (CAIA)	Member Member
Compliance	• Compliance	Member

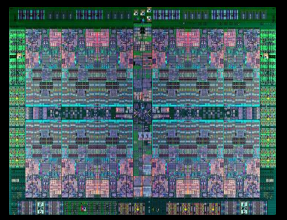
A new Open Power Linux environment

- OpenPOWER is not the traditional Power Linux with a new name
  - Significant discontinuity and fresh start
  - No Binary Compatibility to Legacy Power Linux
    ⇨ new environment “ppc64le”
  - New: Firmware, Hypervisor, data layout, source code, ELFv2 ABI
- What changes for application developers?
  - Byte order
  - New ABI
  - Vector programming API
- Extremely rapid turn-around from decision to develop to deployment
  - First discussions in March 2013, Linux distro builds starting October 2013
  - >40000 packages built within short time

IBM & NVIDIA Accelerating Computing

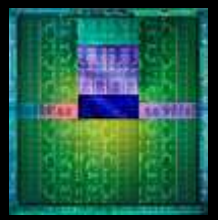
Next-Gen IBM Supercomputers and Enterprise Servers

Long term roadmap integration



POWER CPU

\+



Tesla GPU

Enterprise & Data Analytics Software

Applications, Tools, Algorithms, Libraries, Languages, Compilers

GPU-Accelerated POWER-Based Systems Available in 2014

POWER8 Processor

Technology

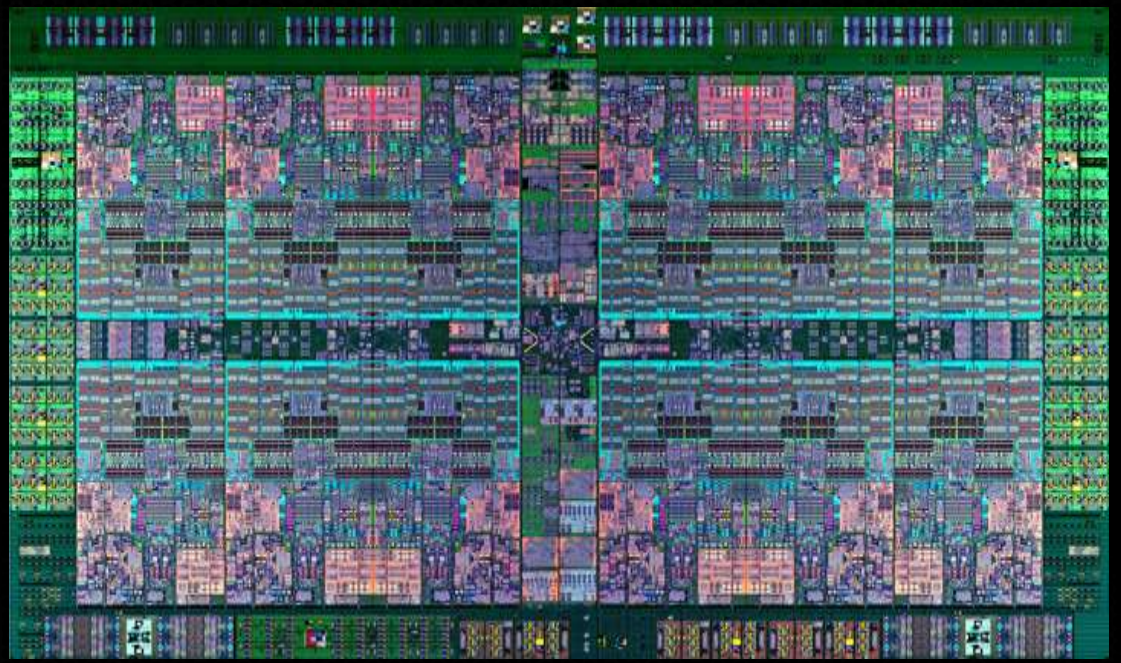
- 22nm SOI, eDRAM, 15 ML 650mm2

Cores

- 12 cores (SMT8)
- 8 dispatch, 10 issue, 16 exec pipe
- 2X internal data flows/queues
- Enhanced prefetching
- 64K data cache,
  32K instruction cache

Accelerators

- Crypto & memory expansion
- Transactional Memory
- VMM assist
- Data Move / VM Mobility



Energy Management

- On-chip Power Management Micro-controller
- Integrated Per-core VRM
- Critical Path Monitors

Caches

- 512 KB SRAM L2 / core
- 96 MB eDRAM shared L3
- Up to 128 MB eDRAM L4
  (off-chip)

Memory

- Up to 230 GB/s
  sustained bandwidth

Bus Interfaces

- Durable open memory attach interface
- Integrated PCIe Gen3
- SMP Interconnect
- CAPI (Coherent Accelerator Processor Interface)

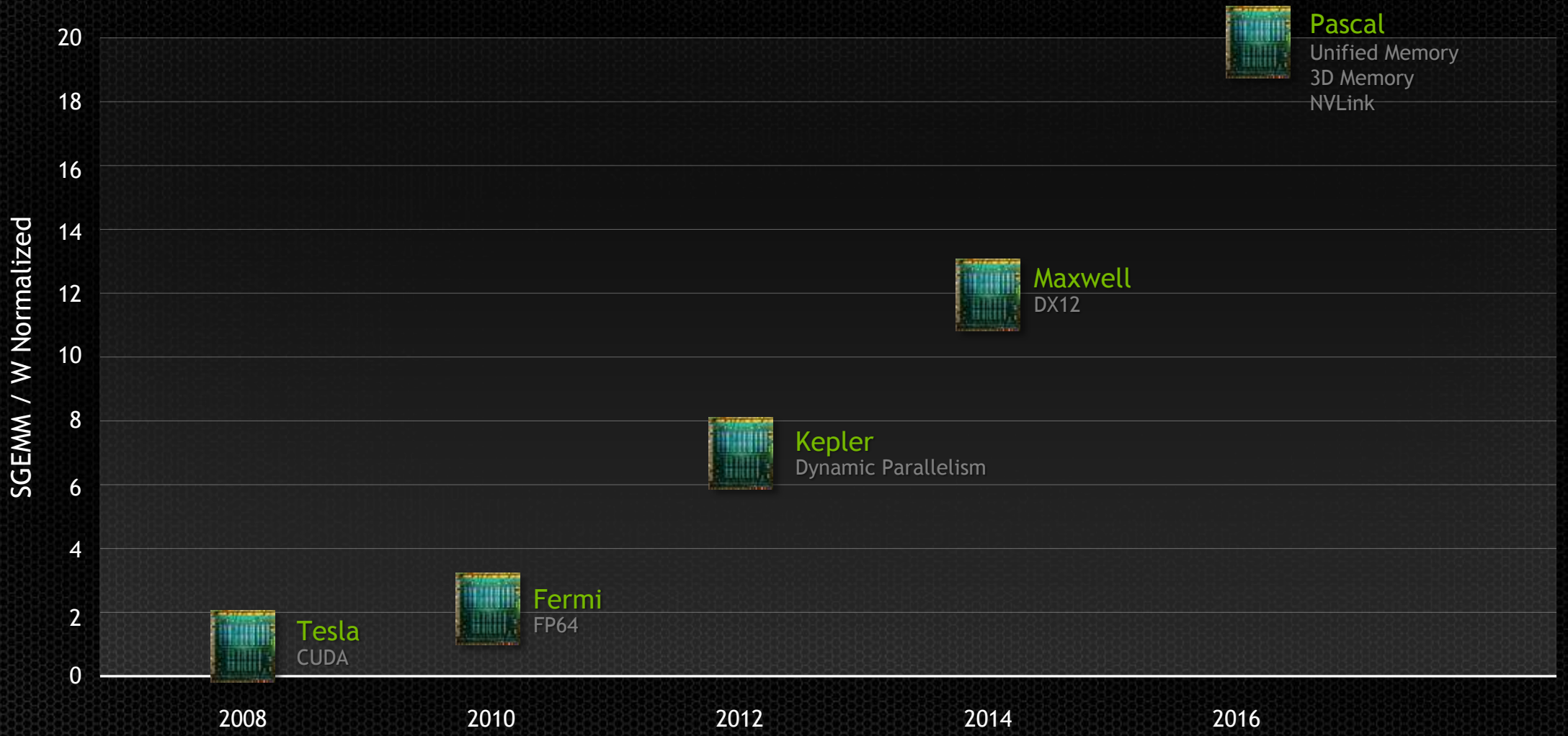
The World's First GPU-accelerated Power8 Server

IBM POWER S824L

- 2x POWER8 CPUs
  - 10 cores 3.425 GHz or
  - 12 cores 3.026 GHz
  - 6MB L2 / 96MB L3 each CPU
- 1TB Memory Capacity
- 384 GB/s Max Mem Bandwidth
- 2 Tesla K40 GPU Accelerators
- Linux (Ubuntu, then RedHat)
- CAPI Support

Available starting Oct 31st!

Strong GPU Roadmap

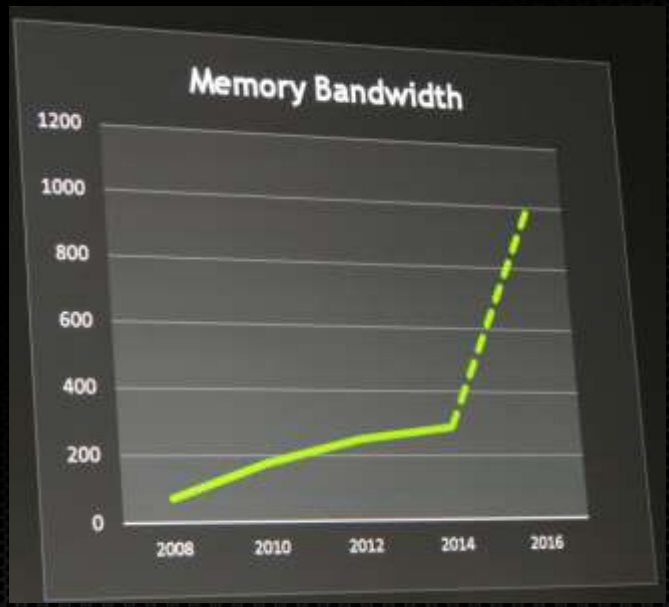


Pascal GPU

- Optimized for double precision FP
- Very high bandwidth, large capacity 3D memory on package
- NVLINK for high **bandwidth** CPU ⇔ GPU and **GPU** ⇔ **GPU** interconnect
- Unified Memory (UM) HW support
- New packaging allows much denser solutions (one-third the size of current PCIe boards)

Stacked Memory

- 3D chip on wafer integration
  - Multiple layers of DRAM components will be integrated vertically on the package along with the GPU
- Compared to GDDR5 memory
  - 4x Higher Bandwidth
  - 3x Larger Capacity
  - 4x More Energy Efficient per bit

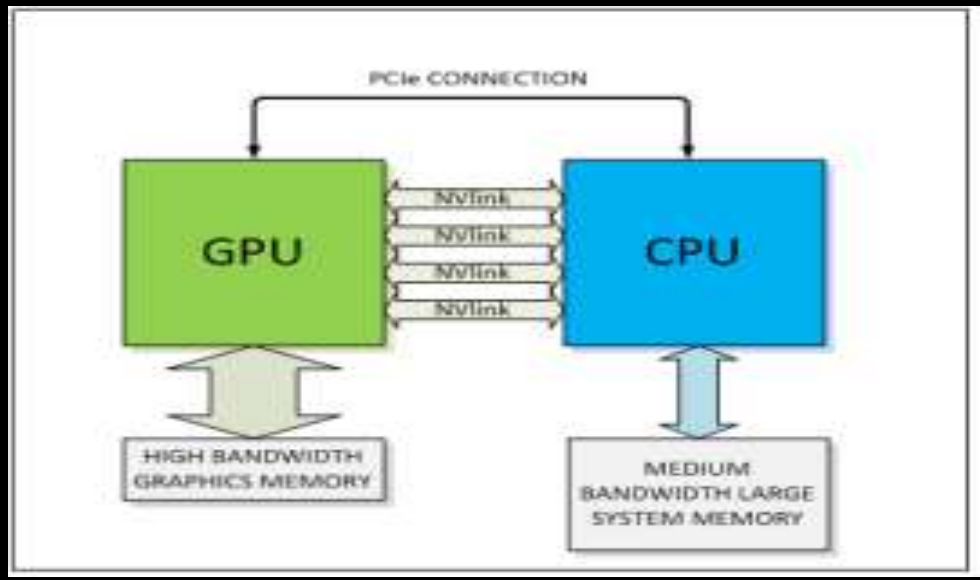


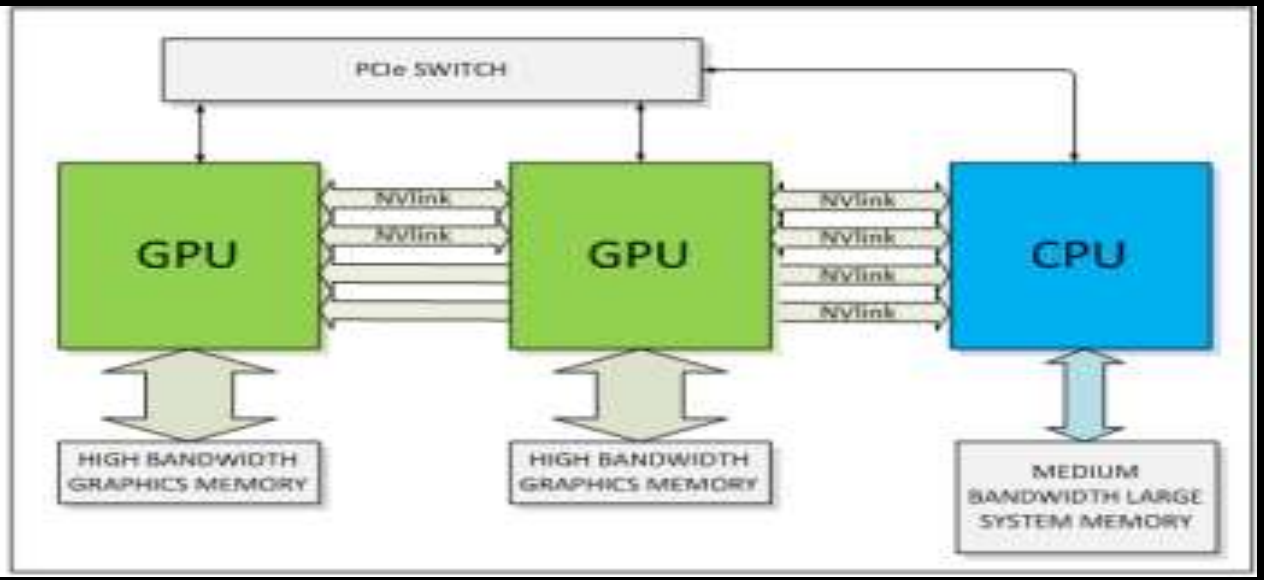
NVLINK

- CPU ⇔ GPU communication limited by low bandwidth connection via PCI-e
- NVLINK is a high speed interconnect between CPU ⇔ GPU and GPU ⇔ GPU
- Basic building block is a 8-lane, differential, dual simplex bidirectional link
- Multiple links can be aggregated to increase BW of a connection
- NVLink will provide between 80 and 200 GB/s of bandwidth
- Preserves the PCIe programming model
  - CPU-initiated transactions such as control and configuration over a PCIe connection
  - GPU-initiated transactions use NVLink
- Allowing the GPU full-bandwidth access to the CPU's memory system
- NVLink is more than twice as energy efficient as a PCIe 3.0 connection

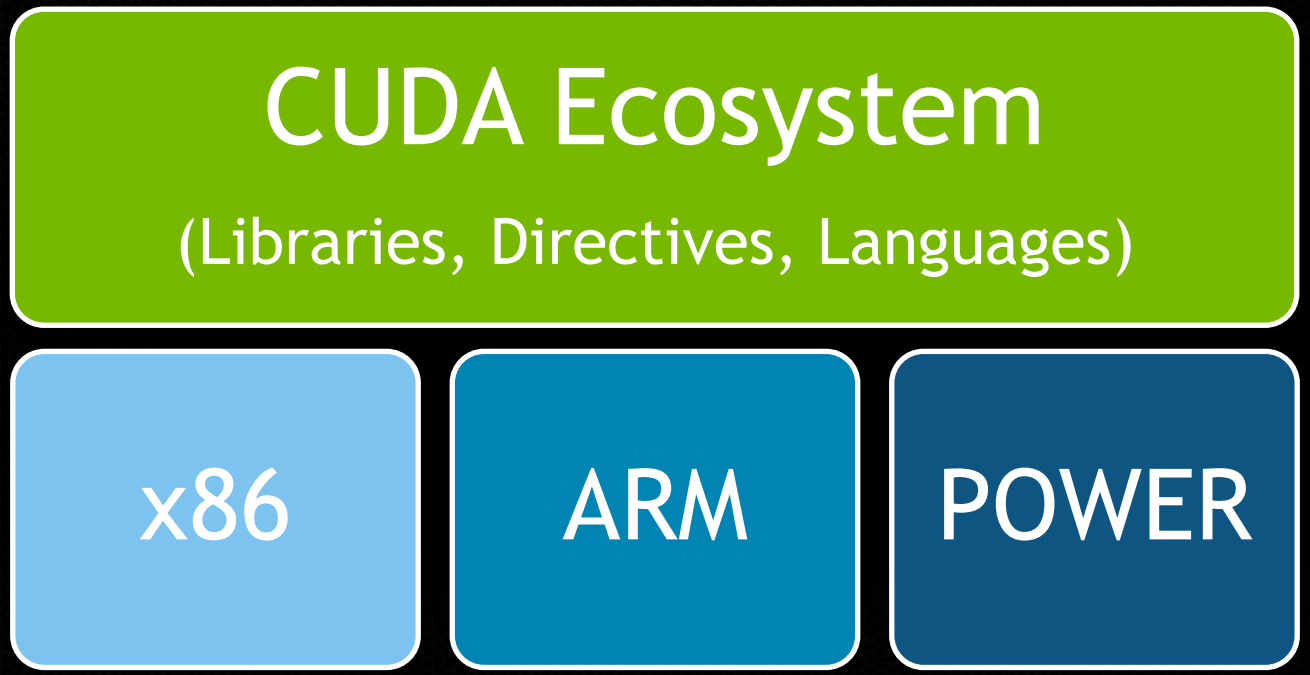
http://devblogs.nvidia.com/parallelforall/nvlink-pascal-stacked-memory-feeding-appetite-big-data/

NVLINK


PCIe CONNECTION
GPU
CPU
NVlink
NVlink
NVlink
NVlink
HIGH BANDWIDTH GRAPHICS MEMORY
MEDIUM BANDWIDTH LARGE SYSTEM MEMORY


PCIe SWITCH
GPU
GPU
CPU
NVlink
NVlink
NVlink
NVlink
NVlink
NVlink
HIGH BANDWIDTH GRAPHICS MEMORY
HIGH BANDWIDTH GRAPHICS MEMORY
MEDIUM BANDWIDTH LARGE SYSTEM MEMORY

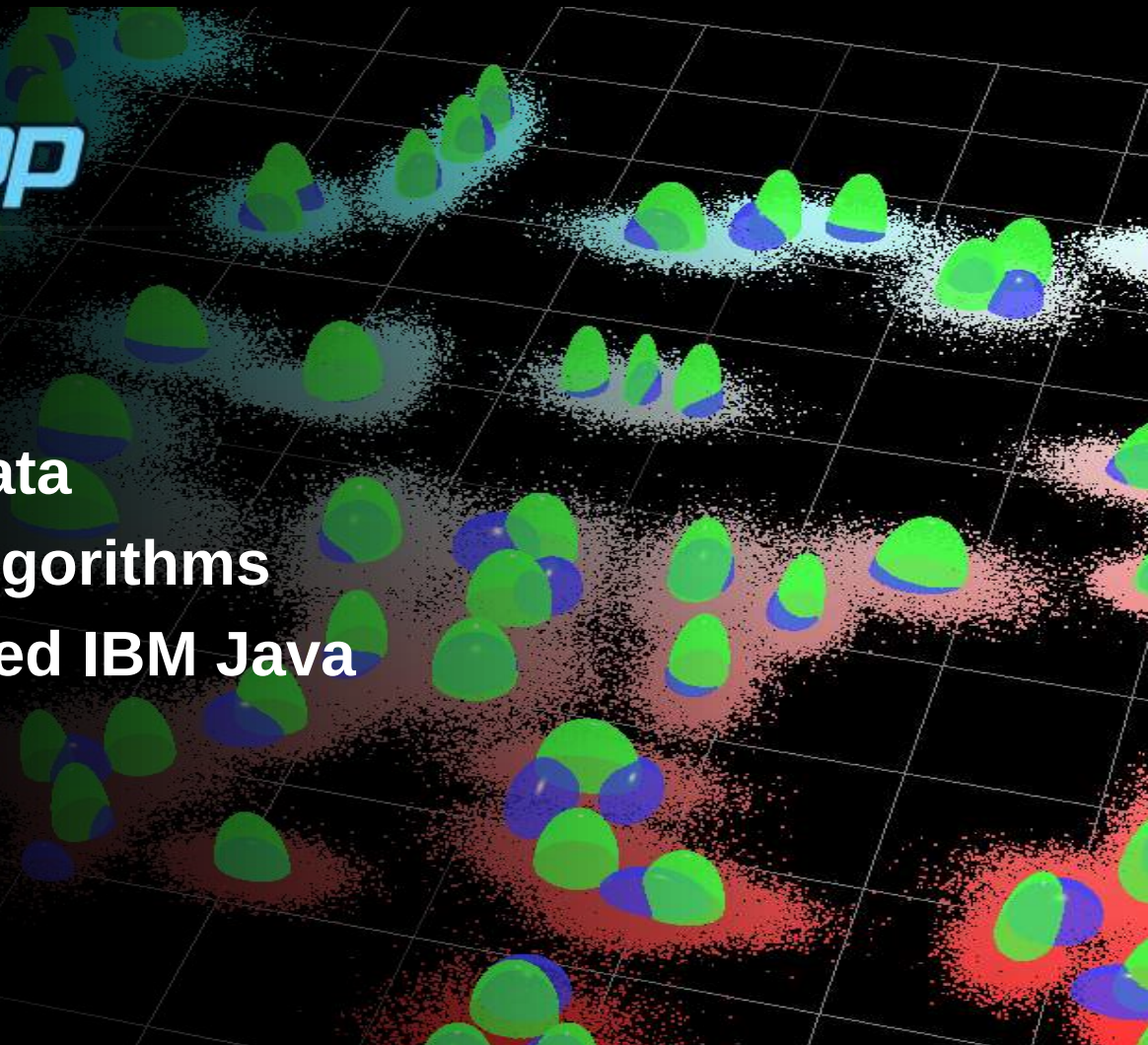
Three ISAs, One Programming Model



Java GPU Acceleration on OpenPOWER

- IBM and NVIDIA partnering to bring GPU acceleration to mainstream Java workloads and enterprise environments on OpenPOWER
- Evolve the Java support to enable both transparent Java acceleration and Java programmer enablement to program explicitly for GPUs from Java
- Key compute-intensive algorithms in Java standard libraries and domain-specific libraries to be accelerated with GPU
- First step was to build a GPU control and runtime framework in Java
  - Allow Java programmers to do data transfers and launch GPU kernels
  - Gain experience in APIs and GPU programming models applied to Java

GPU-Accelerated Hadoop



Extract insights from customer data

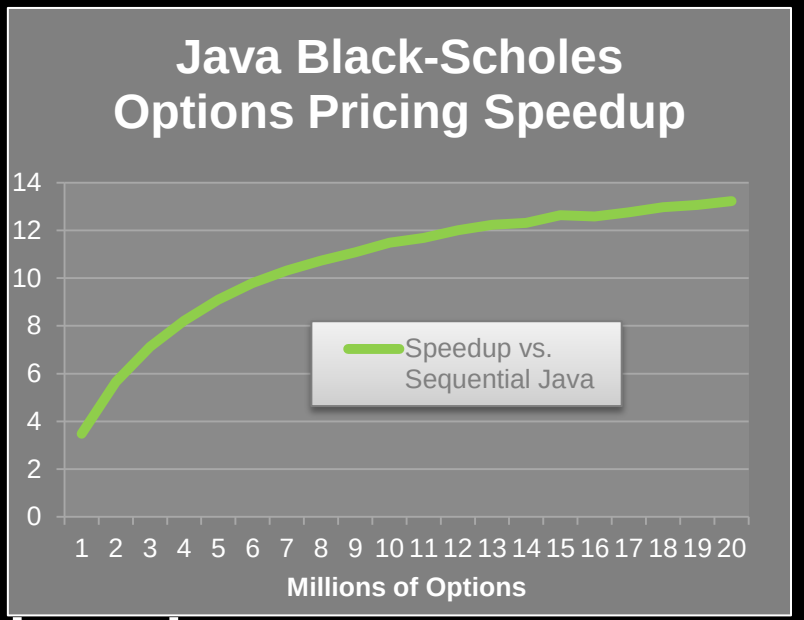
Data Analytics using clustering algorithms

Developed using CUDA-accelerated IBM Java

Compile Java for GPUs

- Approach: apply a closure to a set of arrays

```
// vector addition
float[] X = {1.0, 2.0, 3.0, 4.0, … };
float[] Y = {9.0, 8.1, 7.2, 6.3, … };
float[] Z = {0.0, 0.0, 0.0, 0.0, … };
jog.foreach(X, Y, Z, new jogContext(),
  new jogClosureRet<jogContext>() {
    public float execute(float x, float y) {
      return x + y;
    }
  }
);
```



- foreach iterations parallelized over GPU threads
  - Threads run closure `execute()` method