CERN openlab

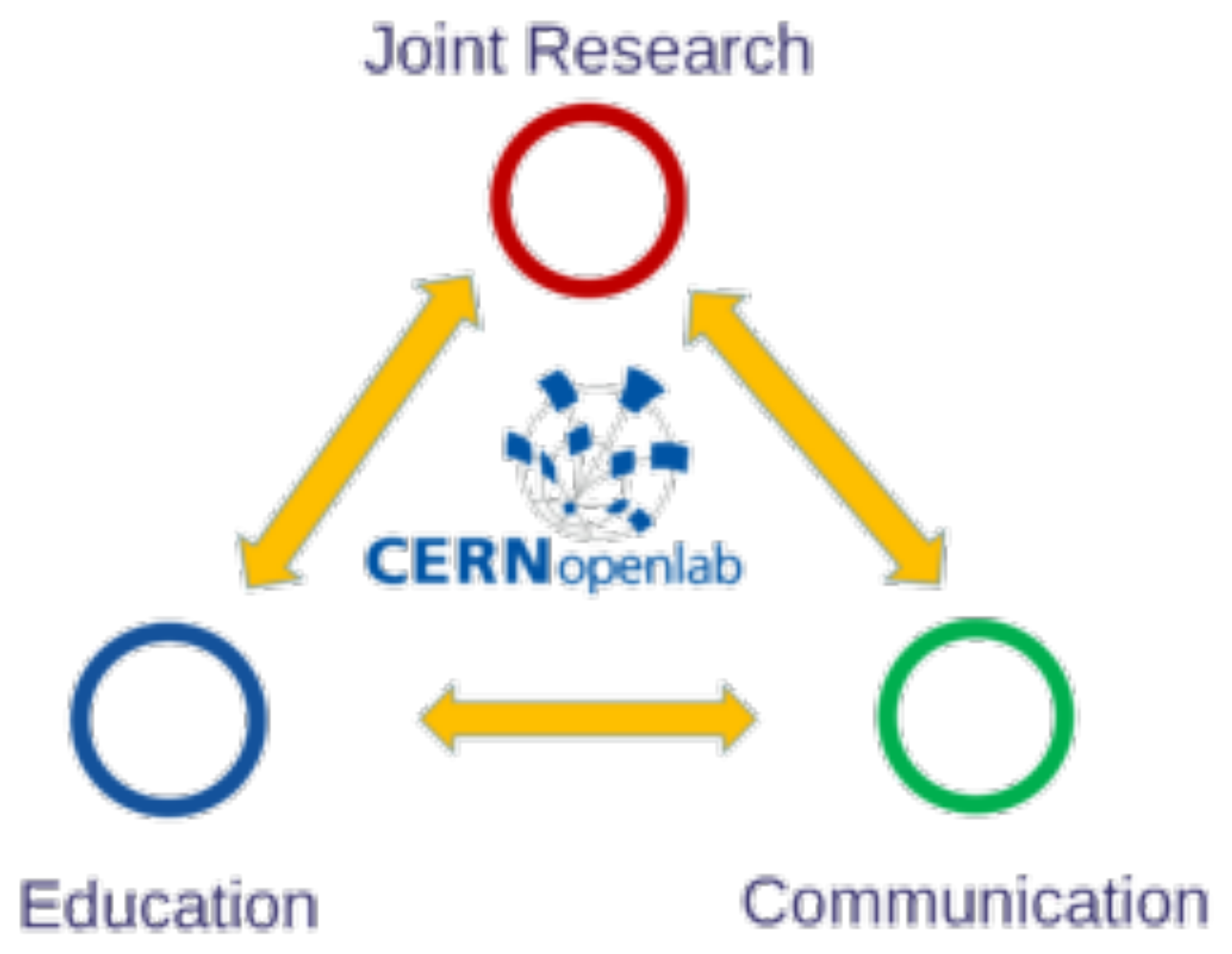
What hardware and computing facilities do we expect in 5-7 years for analysis?

Fons Rademakers, ROOT Founder and CERN openlab Chief Research Officer,

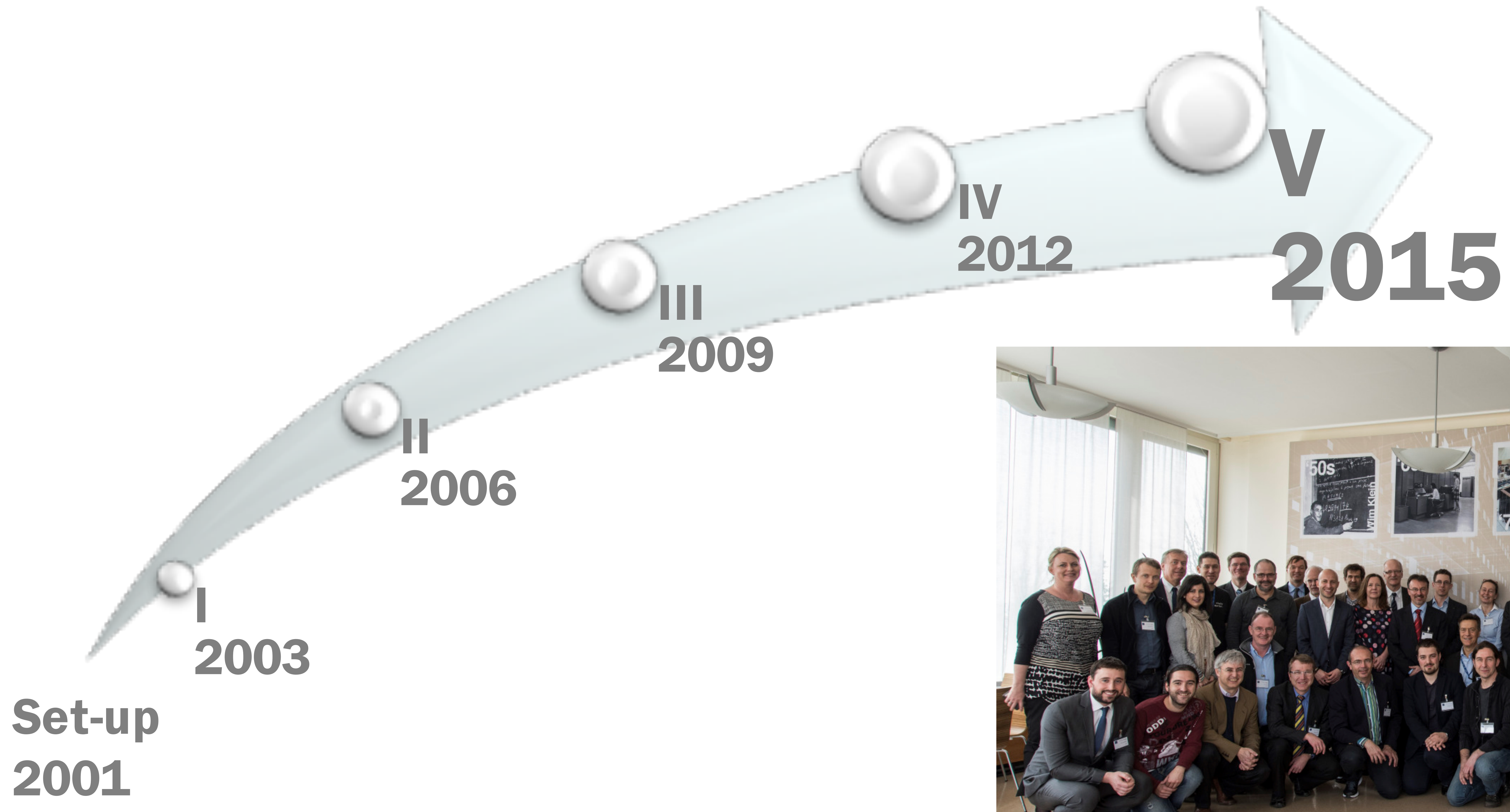
HSF Analysis Ecosystem Workshop, Amsterdam, 22/24—May-2017.

CERN openlab

- CERN openlab, a science – industry partnership to drive R&D in IT
- CERN openlab promotes innovation, education and entrepreneurship in IT
- Working on multi-disciplinary projects exploiting the latest IT techniques
- Development of educational and KT projects
- Dissemination of results



16 Years of Successful Collaborations



15 Years Anniversary

2016

CERN openlab

Current CERN openlab Members

Partners	intel	ORACLE	SIEMENS		
Contributors	rackspace the open cloud company	SEAGATE	CISCO	IDT	BROCADE
Associates	Yandex	ComTrade	HUAWEI		
Research	EMBL-EBI	GSI	Newcastle University	Kazan Federal University	Innopolis University

Back to the Future… 1995

Back to the Future… 1995 Hardware

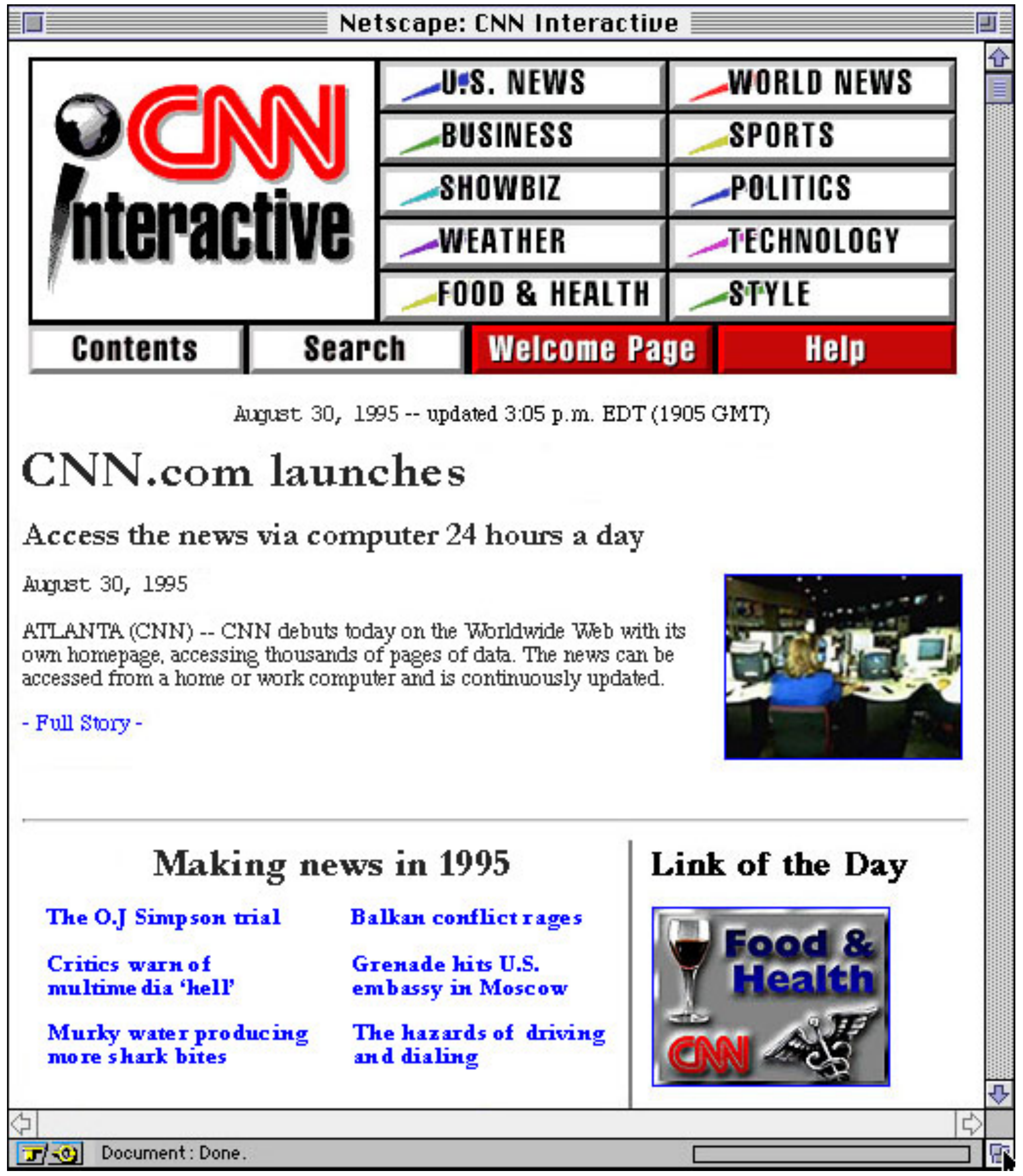
Dell was selling a top-of-the-line 486 with a 66MHz processor, eight megabytes of RAM and a 320-megabyte hard drive for $4,400.

HP workstations running HP-UX on PA-RISC was the original ROOT team development platform. In 1996 replaced by HP PC's running Linux 1.0 on Pentium Pro.

Part	1995 PC	2017 PC
Memory	8mb at $400 per 4 MB	8 gigabytes is common…, which is 8,000 megabytes
Hard Drive	400 to 1,000 megabytes (80MB $300, 1GB $3000)	500 Gigabytes is low end… which is 500,000 megabytes. 500 GB can be as low as $50. Terabyte drives are common (1,000,000 megabytes) for less than the cost of a 1994, 400 MB drive. Depending
Processor	33MHz	4,000 MHz+ with multiple cores (it's very affordable to get 8) and countless optimizations (clock speed is not a clear measurement for processing power)
Video	24-bit accelerated	PCI Express 2 (replacing PCI, and then AGP) with 1-2 Gigabytes of dedicated RAM, for about $250
Monitor	14" CRT	27" wide screen LCD/LED
Sound	Sound Blaster 16 (16-bit)	24-bit, PCI Express, 3d, quad core processors with onboard RAM
Modem	28.8	Obsolete, except for users in very rural areas
Optical Disk	2x CD-ROM	Arguably moving towards obsolescence with digital downloads so prevalent. BluRay, DVD, some CD-ROM remain…

Back to the Future… 1995 Software

- Linux 1.0, WinNT 3.51 and Win95 released
- amazon.com, yahoo.com, eBay.com, AltaVista created
- Java, Javascript, Ruby and PHP released
- C++ 10 years old, but no standard and no STL



August 30, 1995 -- updated 3:05 p.m. EDT (1905 GMT)

CNN.com launches

Access the news via computer 24 hours a day

August 30, 1995

ATLANTA (CNN) -- CNN debuts today on the Worldwide Web with its own homepage, accessing thousands of pages of data. The news can be accessed from a home or work computer and is continuously updated.

- Full Story -

Making news in 1995

The O.J Simpson trial

Balkan conflict rages

Critics warn of multimedia 'hell'

Grenade hits U.S. embassy in Moscow

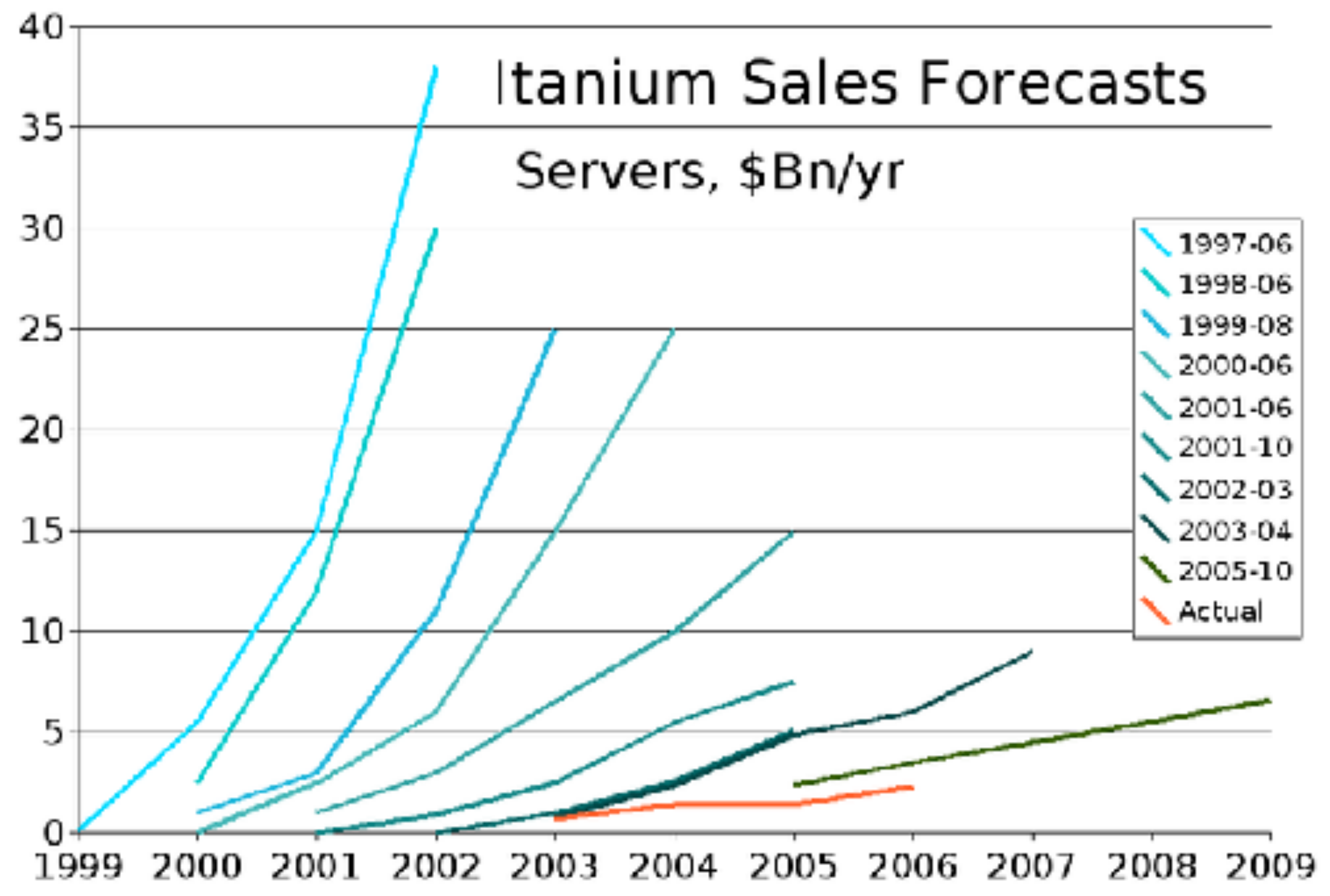
Murky water producing more shark bites

The hazards of driving and dialing

Link of the Day

Back to the Future… 1995 from HEP Perspective

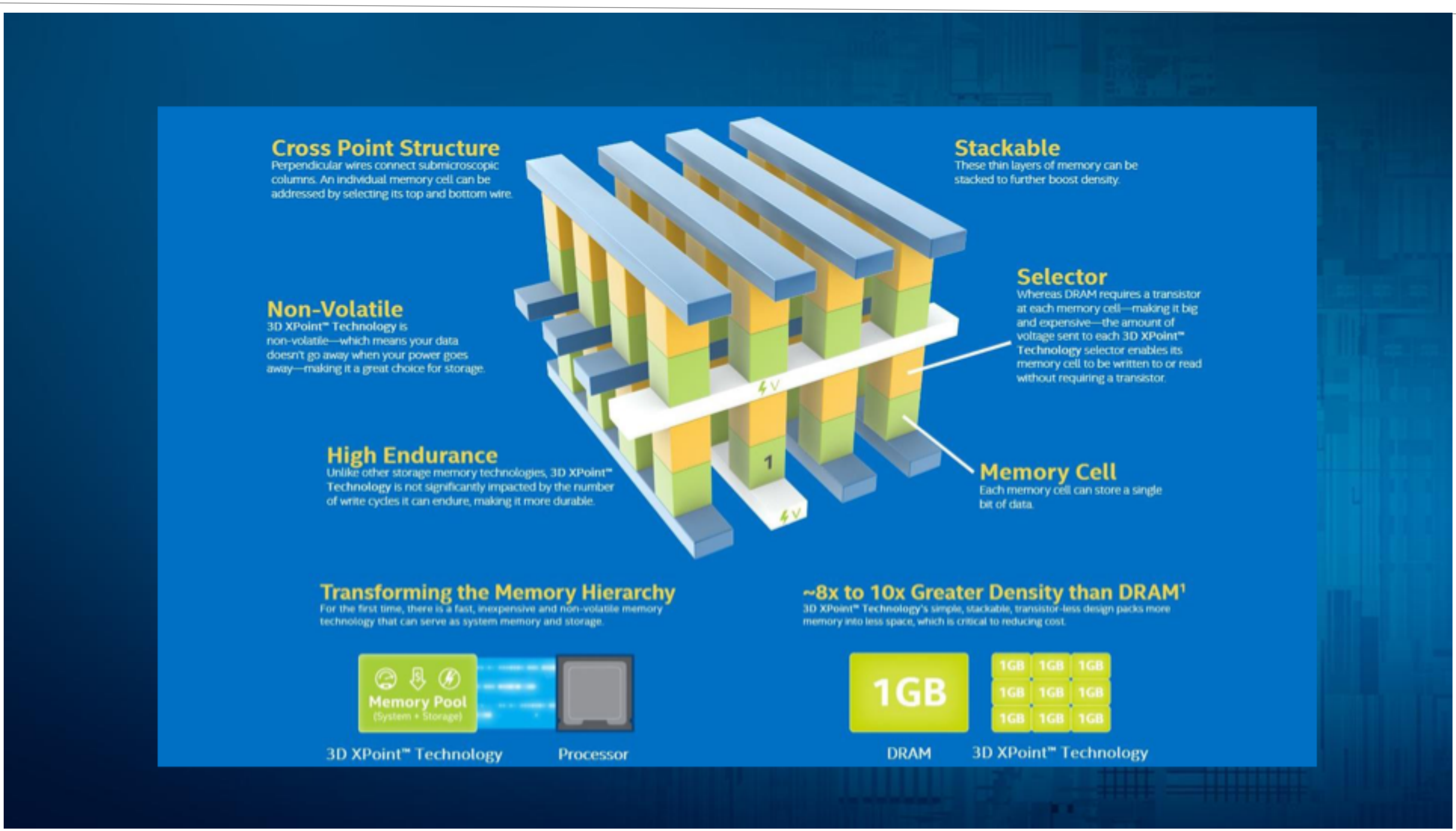
- Hardware
  - Pentium Pro, end of MPP era
  - Future was going to be VLIW (Intel Itanium)



- Software
  - Windows is going to be everywhere (Linux nice but not for serious computing)
  - Data will be stored in a global object oriented Objectivity/DB database
  - Commercial software solutions for math, 2D & 3D graphics, etc. will be acquired and released as LHC++ (software for the LHC era deemed too complex to be developed by physicists. Open source was taking off, but fears about quality and sustainability.)
  - C++ or maybe Eiffel, but surely an OO language (Java, Javascript, PHP, did not yet exist, Python niche language)

The Next 3-5 Years

3D XPoint NVRAM Technology



Cross Point Structure
Perpendicular wires connect submicroscopic columns. An individual memory cell can be addressed by selecting its top and bottom wire.

Non-Volatile
3D XPoint™ Technology is non-volatile—which means your data doesn't go away when your power goes away—making it a great choice for storage.

High Endurance
Unlike other storage memory technologies, 3D XPoint™ Technology is not significantly impacted by the number of write cycles it can endure, making it more durable.

Stackable
These thin layers of memory can be stacked to further boost density.

Selector
Whereas DRAM requires a transistor at each memory cell—making it big and expensive—the amount of voltage sent to each 3D XPoint™ Technology selector enables its memory cell to be written to or read without requiring a transistor.

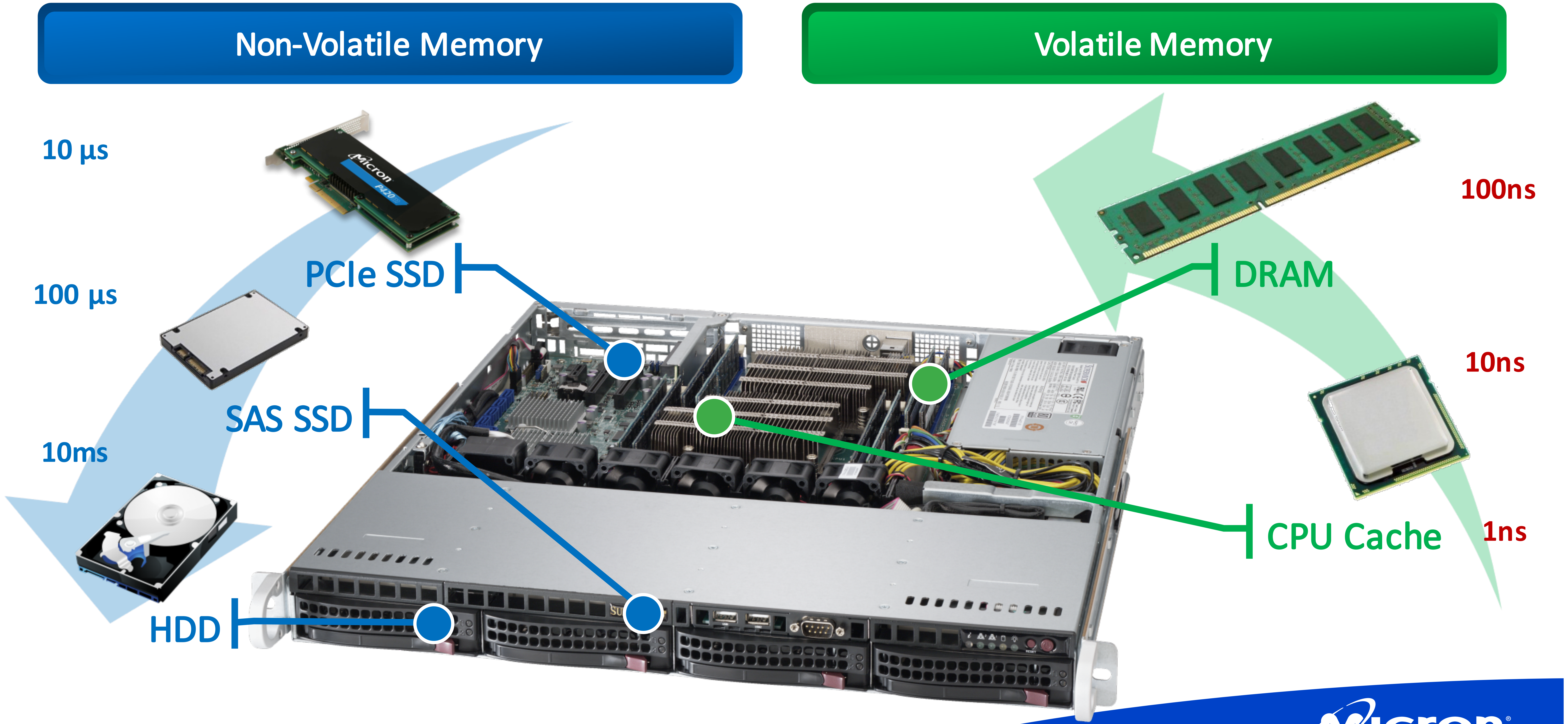
Memory Cell
Each memory cell can store a single bit of data.

Transforming the Memory Hierarchy
For the first time, there is a fast, inexpensive and non-volatile memory technology that can serve as system memory and storage.

~8x to 10x Greater Density than DRAM[1]
3D XPoint™ Technology's simple, stackable, transistor-less design packs more memory into less space, which is critical to reducing cost.

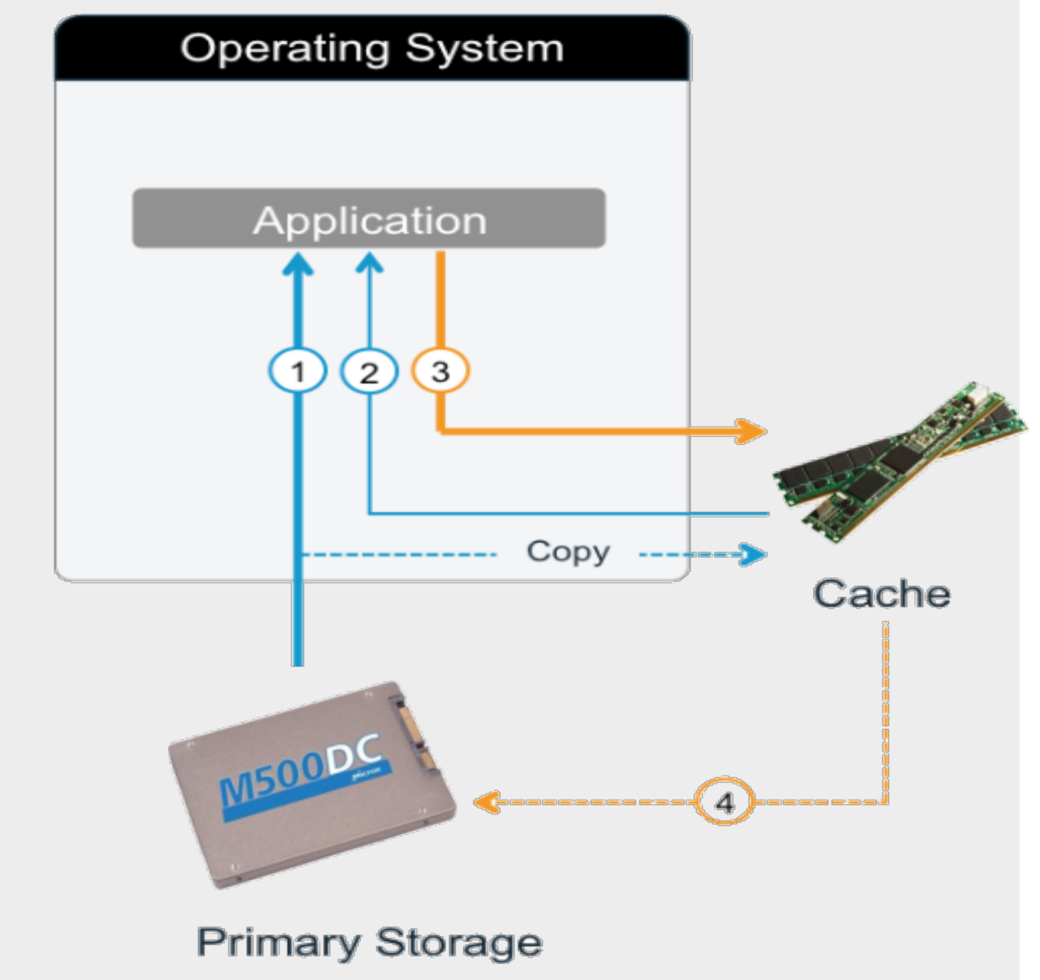
The Challenge: Nonvolatile Memory Latency

- As CPU technology scales, memory IO creates significant performance bottlenecks
- Huge latency gap in memory hierarchy between volatile and non-volatile technologies
- Latency gap widens with the introduction of DDR4



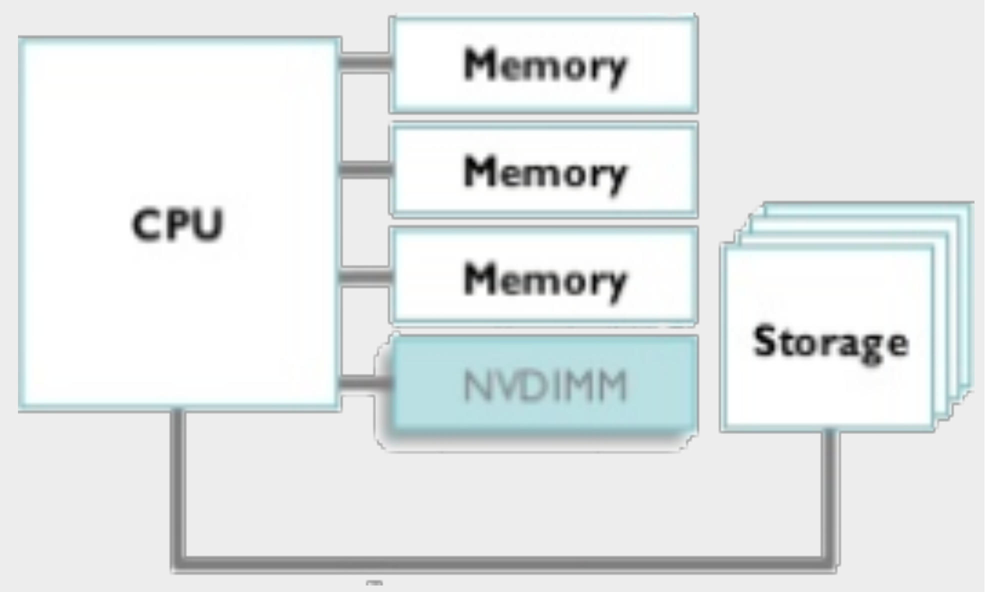
Use Cases and Persistent Variables

Case #1: Write Caching For MLC SSDs



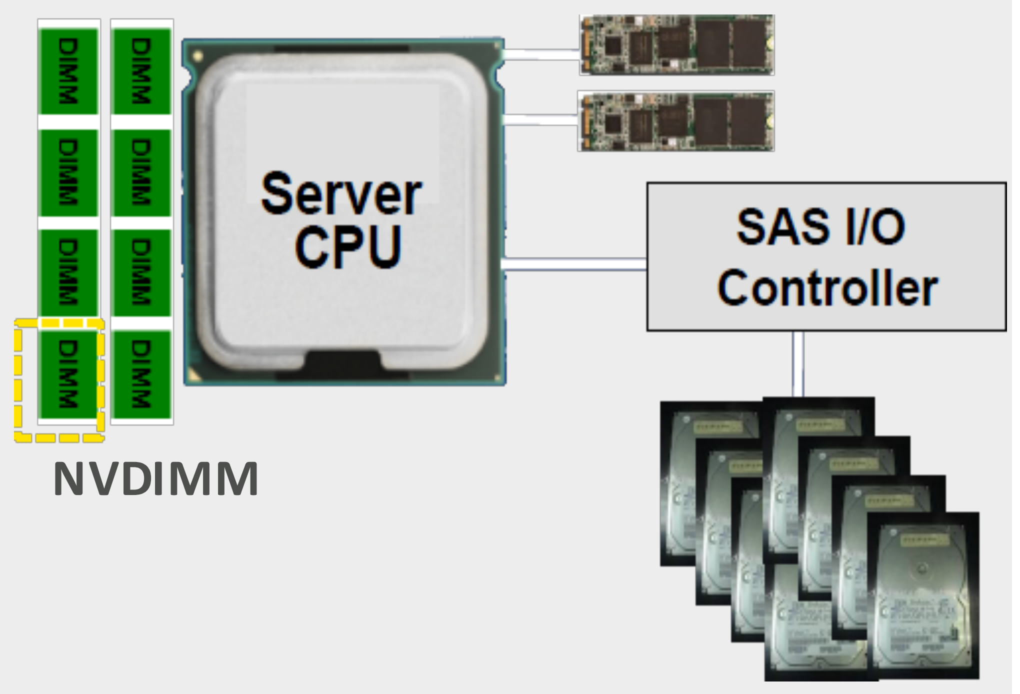
- Extends life of SSD and improves performance for write-intensive apps

Case #2: Low Write Latency Persistent Storage



- >100x lower write latency versus PCIe SSD w/ unlimited endurance

Case #3: Unified Open Software-Defined Server RAID



- Scalable unified RAID performance for SSDs and HDDs

Persistent variables: Metadata, Checkpoint State, Host Caching, RAMDisk, RAID Compute, Write Buffer, SSD Mapping, Journaling, Logging

Micron

3D XPoint Game Changer

- The x’es:
  - 1000x lower latency than NAND SSD
  - 10x denser than DRAM
  - 1000x durability of NAND
  - 1x cost of DRAM

3D XPoint Game Changer

- Forces rethinking of several areas in the software stack:
  - On the OS level:
    - No more need for disk buffer caches
    - File access, still via Posix and/or DMA or some new protocol
  - On the program and framework level:
    - No more need for caching or read-ahead
    - Different ROOT file layout to allow direct mapping of branches
    - Storage of run-time caches (xrootd caches, EOS name server tables)
  - Programming languages:
    - Even Java can become a bottleneck as the I/O latency disappears (Apache Big Data stack)

Many-Core Co-Processors and Code Modernization

- Intel Xeon-Phi (KNL) co-processor
- NVidia GPU’s
- Both offer a lot of potential already for quite some years
- Current generations are making a jump in performance
- Especially now that our software and frameworks become multi-core aware we can start reap the benefits

A Very Successful Example of Multi-Core Speedup

- A 700 line kernel from a brain cell growth simulation program was optimised for a coding competition
- This kernel took 45 hours to run with the target set of parameters on a single KNC 7120A CPU
- The winner achieved a running time of 8m24 using all 61 cores of the KNC
- A speedup of 320x

From 45 hours
to…..

8 minutes
24 seconds

320x Increase

The Code Changes and Optimisations

- Custom memory allocator, reuse memory for many small memory allocations
- Change from AoS to SoA to allow vectorisation and improved cache layout
- Use OpenMP for parallelisation over all Xeon-Phi cores
- Use icc Cilk+ scatter/gather intrinsics

AoS to SoA

```
// create 3D concentration matrix
float**** Conc;
Conc = new float***[L];
for (i1 = 0; i1 < 2; i1++) {
    Conc[i1] = new float**[L];
    for (i2 = 0; i2 < L; i2++) {
        Conc[i1][i2] = new float*[L];
        for (i3 = 0; i3 < L; i3++) {
            Conc[i1][i2][i3] = new float[L];
            for (i4 = 0; i4 < L; i4++) {
                Conc[i1][i2][i3][i4] = zeroFloat;
            }
        }
    }
}
```

```
// create 3D concentration matrix
float* Conc;
Conc = new float[L*L*L*2];
void* tempMemory=malloc(std::max(L*L*L*2*sizeof(float),finalNumberCells*3*sizeof(int)));
float* tempConc=(float*)tempMemory;
memset(Conc,0,L*L*L*2*sizeof(float));
```

```
for (c=0; c<n; c++) {
    posAll[c][0] = posAll[c][0]+currMov[c][0];
    posAll[c][1] = posAll[c][1]+currMov[c][1];
    posAll[c][2] = posAll[c][2]+currMov[c][2];

    // boundary conditions: cells can not move out of the cube [0,1]^3
    for (d=0; d<3; d++) {
        if (posAll[c][d]<0) {posAll[c][d]=0;}
        if (posAll[c][d]>1) {posAll[c][d]=1;}
    }
}
```

```
for (int c=0; c<n*3; c++) {
    posAll[c] = posAll[c]+currMov[c];

    // boundary conditions: cells can not move out of the cube [0,1]^3
    if (posAll[c]<0) {posAll[c]=0;}
    if (posAll[c]>1) {posAll[c]=1;}
}
```

OpenMP

```
for (int c=0; c<n*3; c++) {
    posAll[c] = posAll[c]+currMov[c];

    // boundary conditions: cells can not move out of the cube [0,1]^3
    if (posAll[c]<0) {posAll[c]=0;}
    if (posAll[c]>1) {posAll[c]=1;}
}
```

```
#pragma omp parallel for simd default(none) firstprivate(posAll,n,currMov)
for (int c=0; c<n*3; c++) {
    posAll[c] = posAll[c]+currMov[c];

    // boundary conditions: cells can not move out of the cube [0,1]^3
    if (posAll[c]<0) {posAll[c]=0;}
    if (posAll[c]>1) {posAll[c]=1;}
}
```

```
for (int c=0; c< n; c+=size) {
    int i[size*3];
    int localSize=MIN(size,(n-c))*3;
    for (int k = 0; k < localSize; k++)
       i[k] = std::min((int)floor(posAll[c*3+k]/sideLength),(L-1));
    for (int j = 0; j < localSize/3; ++j) {
        int position = position((-typesAll[c+j]+1)/2,i[j*3+0],i[j*3+1],i[j*3+2],2,L,L,L);
        Conc[position]=MIN(Conc[position]+0.1,1.f);
    }
}
```

Bulk of the speedup due to OpenMP

```
#pragma omp parallel for default(none) firstprivate(size,posAll,sideLength,L,n,Conc,typesAll)
for (int c=0; c< n; c+=size) {
    int i[size*3];
    int localSize=MIN(size,(n-c))*3;
    for (int k = 0; k < localSize; k++)
       i[k] = std::min((int)floor(posAll[c*3+k]/sideLength),(L-1));
    #pragma omp simd
    for (int j = 0; j < localSize/3; ++j) {
        int position = position((-typesAll[c+j]+1)/2,i[j*3+0],i[j*3+1],i[j*3+2],2,L,L,L);
        Conc[position]=MIN(Conc[position]+0.1,1.f);
    }
}
```

Intel icc Cilk+

```
#pragma omp parallel for default(none) firstprivate(size,posAll,sideLength,L,n,Conc,typesAll)
for (int c=0; c< n; c+=size) {
    int i[size*3];
    int localSize=MIN(size,(n-c))*3;
    for (int k = 0; k < localSize; k++)
       i[k] = std::min((int)floor(posAll[c*3+k]/sideLength),(L-1));
    #pragma omp simd
    for (int j = 0; j < localSize/3; ++j) {
        int position = position((-typesAll[c+j]+1)/2,i[j*3+0],i[j*3+1],i[j*3+2],2,L,L,L);
        Conc[position]=MIN(Conc[position]+0.1,1.f);
    }
}
```

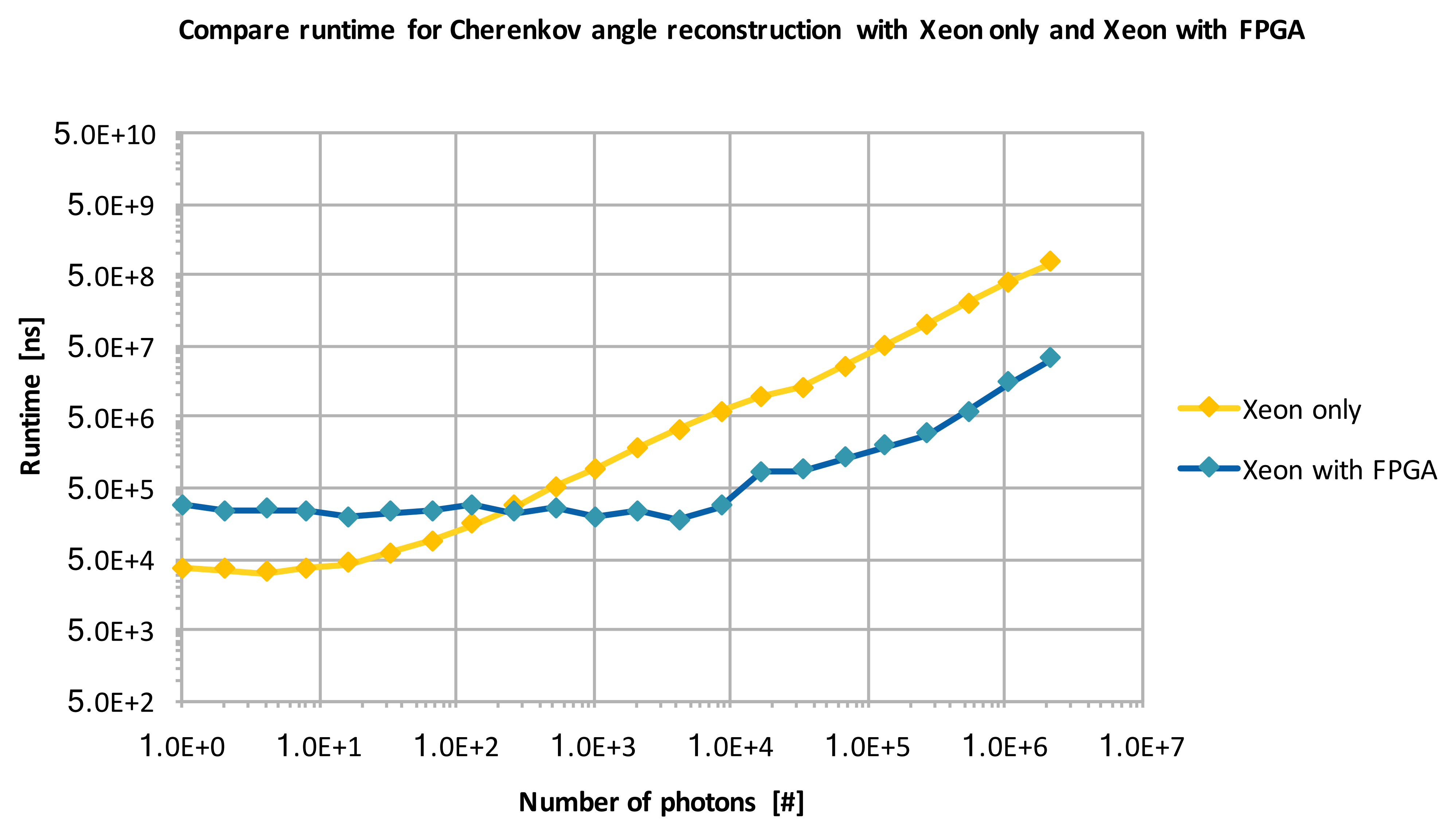
Cilk+ added another 10% speedup

```
#pragma omp parallel for default(none) firstprivate(size,posAll,sideLength,L,n,Conc,typesAll)
for (int c=0; c< n; c+=size) {
    int i[size*3];
    int localSize=MIN(size,(n-c))*3;
#ifdef USE_CILK
    i[0:localSize] = std::min((int)floor(posAll[c*3:localSize]/sideLength),(L-1));
#else
    for (int k = 0; k < localSize; k++)
       i[k] = std::min((int)floor(posAll[c*3+k]/sideLength),(L-1));
#endif
    #pragma omp simd
    for (int j = 0; j < localSize/3; ++j) {
        int position = position((-typesAll[c+j]+1)/2,i[j*3+0],i[j*3+1],i[j*3+2],2,L,L,L);
        Conc[position]=MIN(Conc[position]+0.1,1.f);
    }
}
```

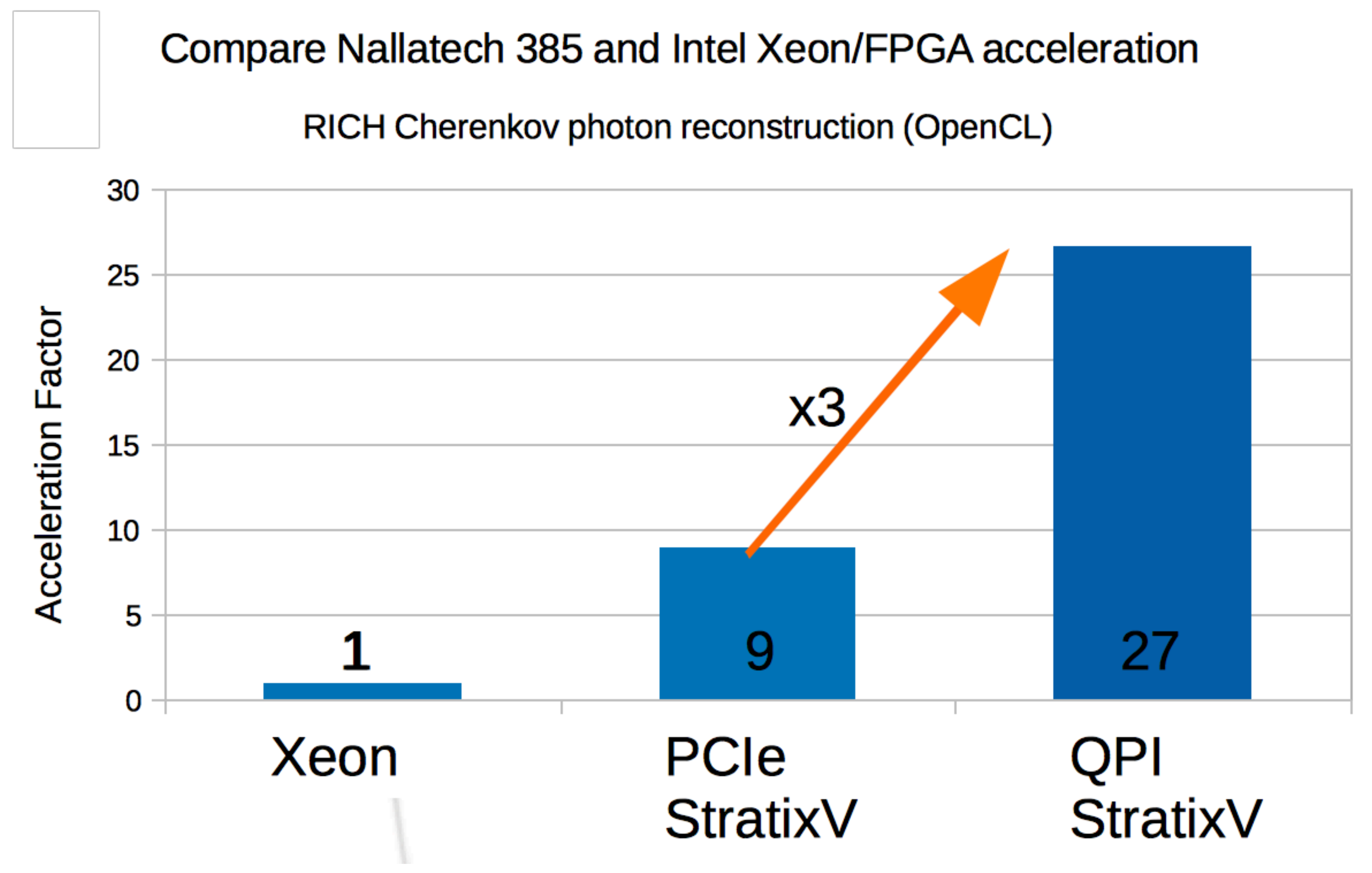
Hybrid Xeon with integrated FPGA

- Xeon/FPGA have an advantage over PCIe based accelerators (both FPGA and GPGPU) because of the cache-coherent, low-latency access to main-memory and CPU (no PCIe bottle-neck) using QPI

- More applicable in TDAQ environments, tuned to a specific task



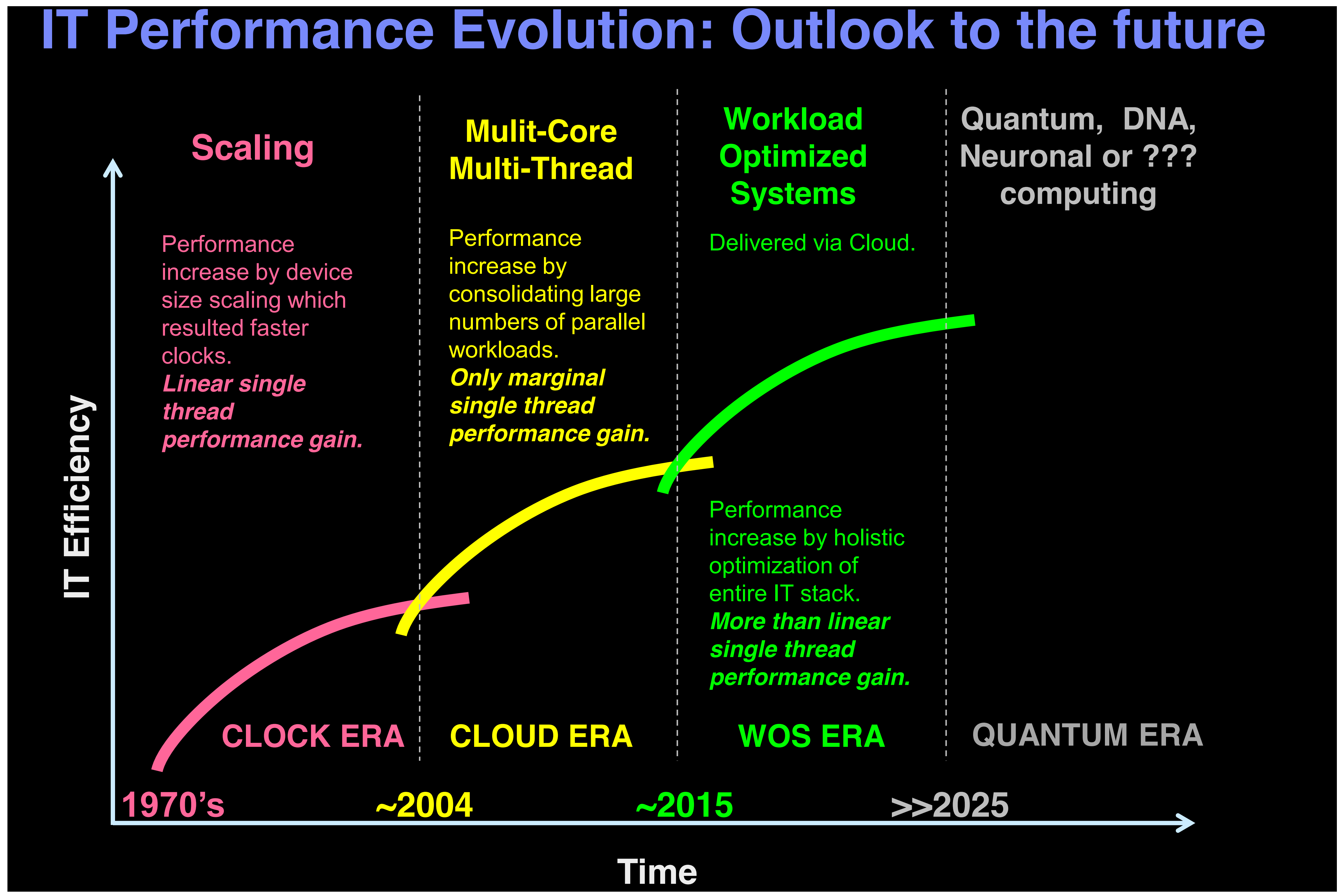
Acceleration of factor up to 35 with Xeon/FPGA



Courtesy of the HTCC project,
for more see talk 13-Oct, 14:15 in Sierra A

Hybrid CPU's

- Die's can contain ever more transistors, but instead of adding more cores, hybrid processing units can be added for specific tasks, like encryption, compression, de-compression, random numbers, etc. Intel and IBM are working on such CPU's. Especially Intel asks our input on what functions should be hardware assisted.



Courtesy IBM

Software in the Next Years

- Typescript and Electron and WebGL are exciting technologies for a new ROOT 2D/3D graphics and GUI back-end

- Investigate new languages like Swift (still needing a Swift/C++ bridge) designed with a REPL from the ground up

Soapbox

- ROOT I/O most feature rich and still fastest solution on the market… but it’s old, so lets see if we can replace it by something fashionable from the Apache Big Data stack that students might have heard about

- Our analyses are mostly I/O bound, still…

- Great, so using a very slow language is not a problem

- But lets vectorise and parallelise and use GPU’s to speed up the processing

- Analysis is mostly done on private resources so it’s ok to waste as many cycles and as much time as you want

Soapbox

- ROOT is only widely used in our community, how about trying to position and advertise its core components as a valid Big Data tool? Ideal job for the HSF.

- HEP works with very long running programs and hence the software needs to be supported for very long periods. This causes problems for adopting commercial components as experiments run often longer than most companies, the same for OS projects, but at least there we've access to the codes. Hence, since long CERN/FNAL/… provide a sustainable software development environment that allows for the long term support, maintenance and continued development of core tools like CERNlib, ROOT and Geant.

- This is fairly unique in science. Our colleagues in e.g. life sciences would love such a situation.

- People will always vote with their feet, not follow committee directives.

Soapbox

- Popular Apache Big Data tools and HPC are not well matched. Java is not an ideal HPC environment. There is now a realisation that for the next level of scaling these tools need to be written in a HPC friendly language, like C++.

- Efforts are underway, like the Apache Kudu storage layer now written in C++:

> "Kudu is specifically designed for use cases that require fast analytics on fast (rapidly changing) data. Engineered to take advantage of next-generation hardware and in-memory processing, Kudu lowers query latency significantly for Apache Impala (incubating) and Apache Spark (initially, with other execution engines to come)." [kudu.apache.org]

- The grass is not always greener on the other side.

Conclusions

- CERN openlab, a science–industry partnership that drives R&D and innovation
- A number of very interesting, high potential, new technologies on the horizon
- Several game changing technologies being researched in the openlab context
- Very interesting times, indeed…

EXECUTIVE CONTACT

Alberto Di Meglio, CERN openlab Head
alberto.di.meglio@cern.ch

TECHNICAL CONTACTS

Maria Girone, CERN openlab Chief Technology Officer
maria.girone@cern.ch

Fons Rademakers, CERN openlab Chief Research Officer
fons.rademakers@cern.ch

COMMUNICATION CONTACT

Andrew Purcell, CERN openlab Communications Officer
andrew.purcell@cern.ch

ADMIN CONTACT

Kristina Gunne, CERN openlab Administration Officer
kristina.gunne@cern.ch