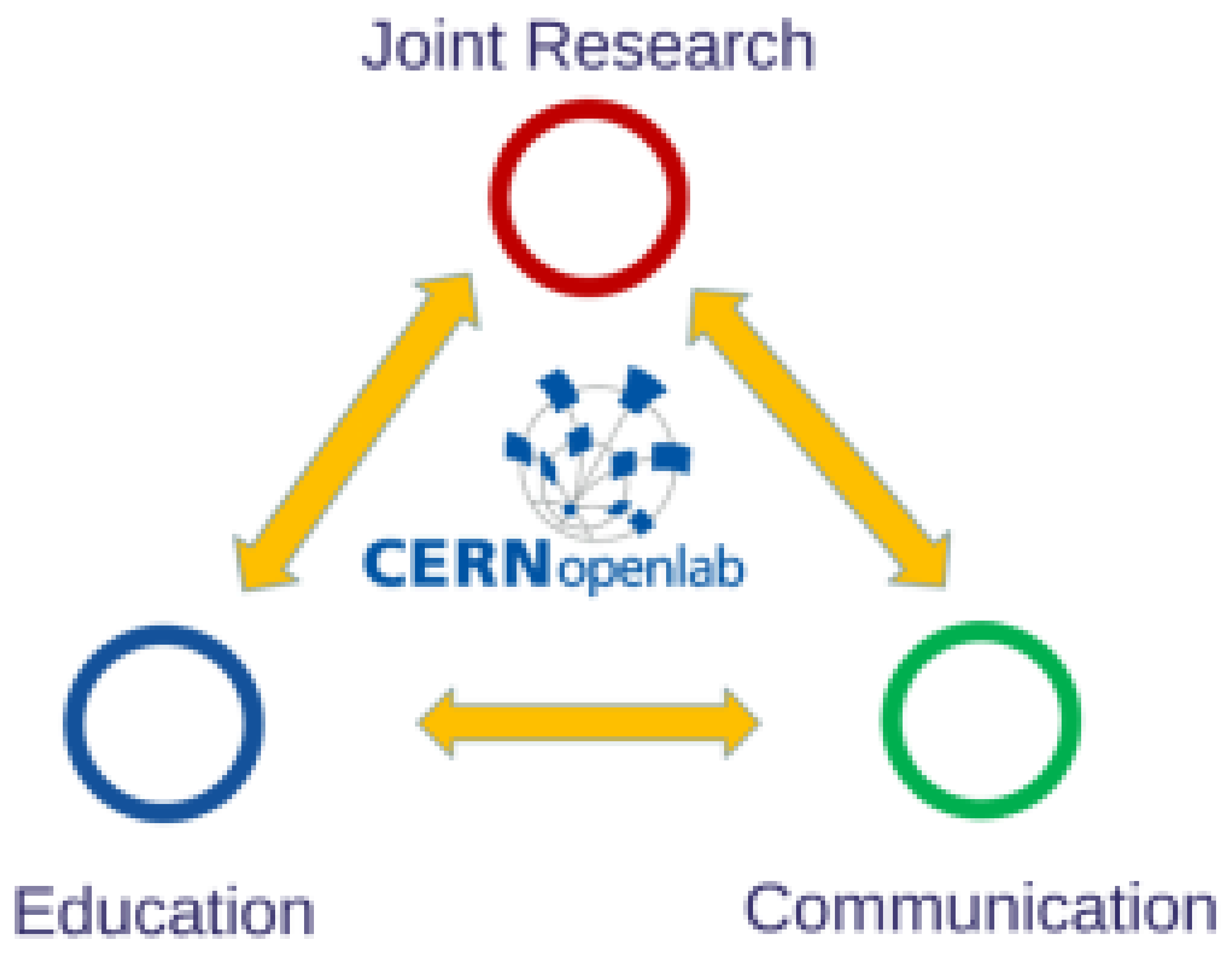
CERN openlab Researched Technologies That Might Become Game Changers in Software Development

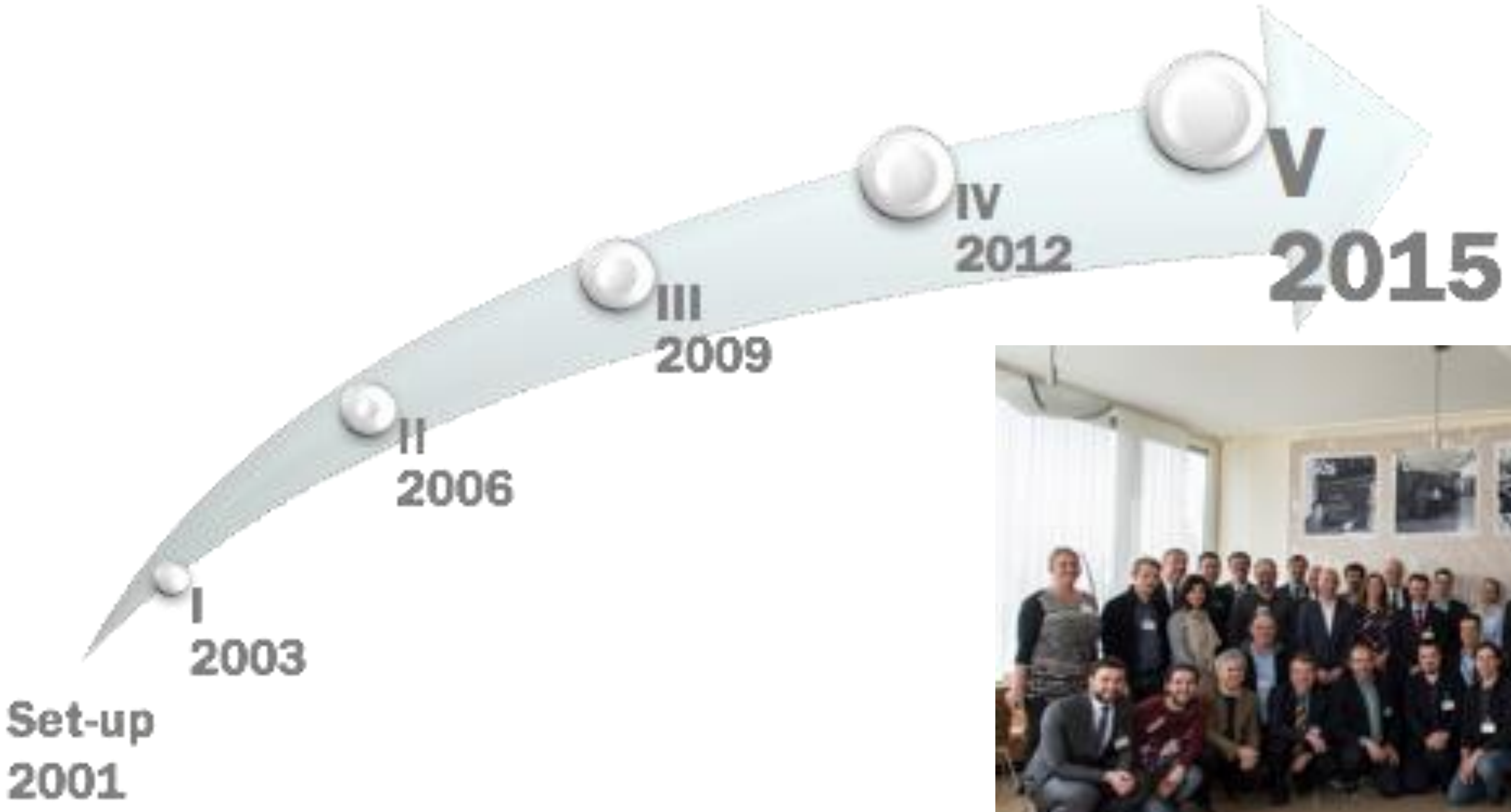
Fons Rademakers, CERN openlab Chief Research Officer
CHEP’16, San Fransisco, 12-Oct-2016.

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



15 Years of Successful Collaborations

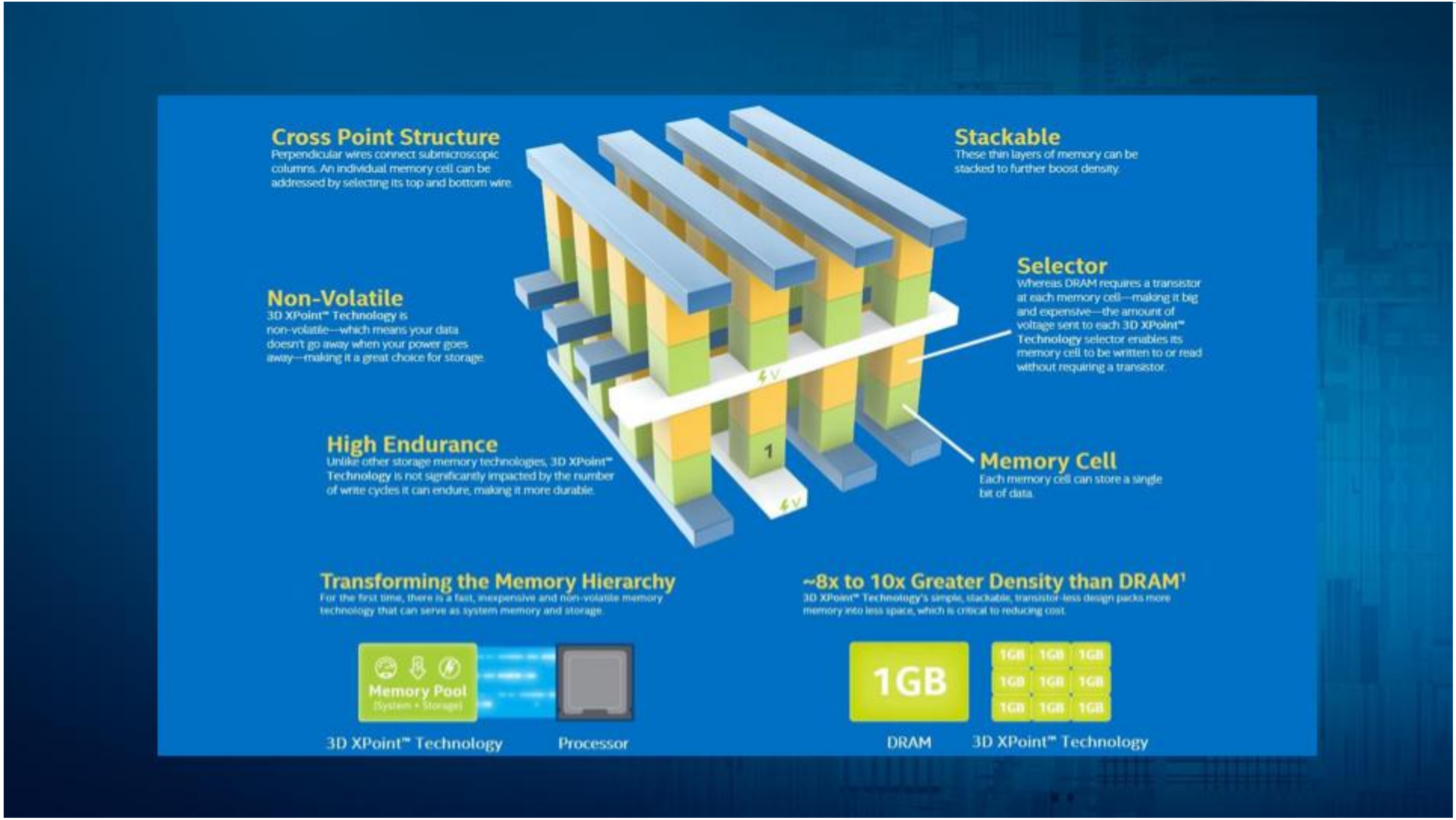


Current CERN openlab Members

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

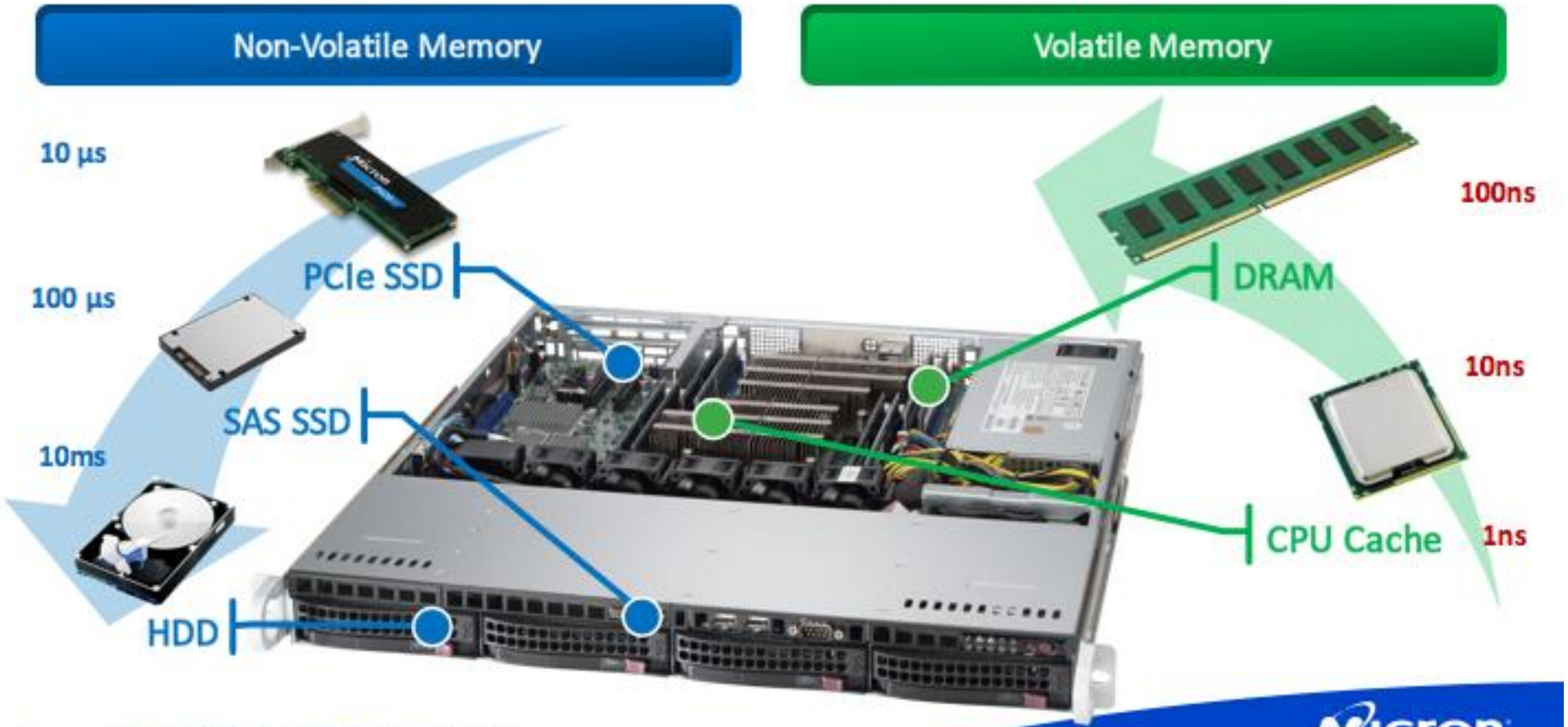
Game Changing Technologies

3D XPoint NVRAM Technology



The Challenge: Nonvolatile Memory Latency

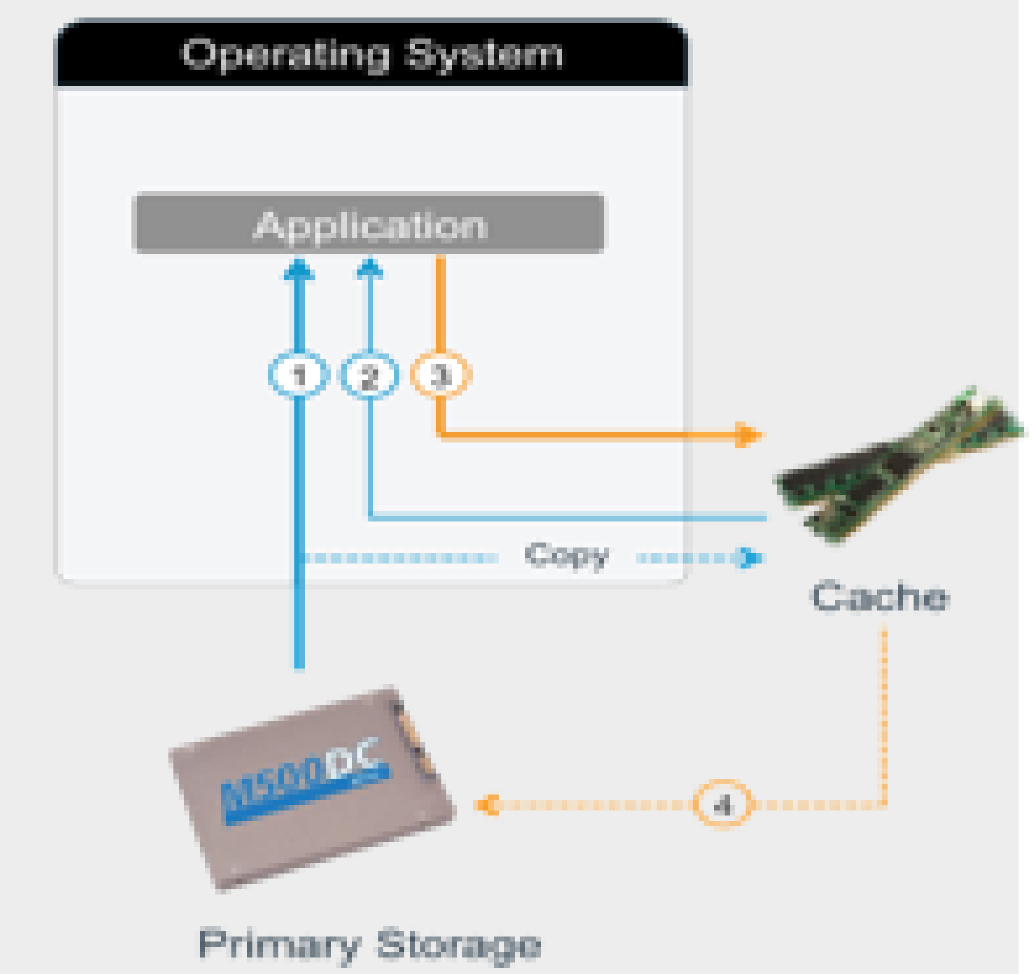
- As CPU technology scales, memory IO creates significant performance bottlenecks
- Huge latency gap in memory hierarchy between volatile and non-volatile technologies
- Latency gap widens with the introduction of DDR4



Micron

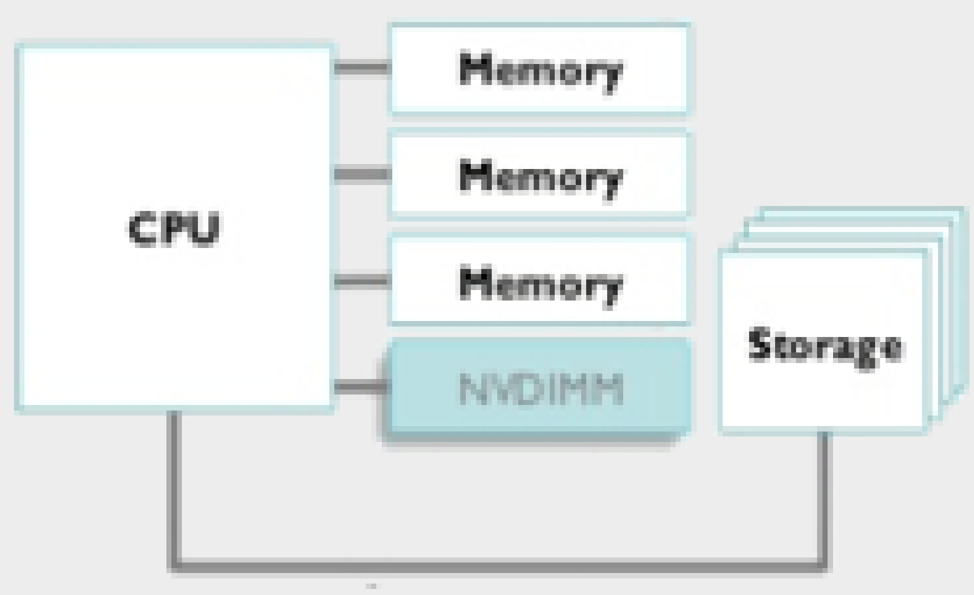
Use Cases and Persistent Variables

Case #1: Write Caching For MLC SSDs



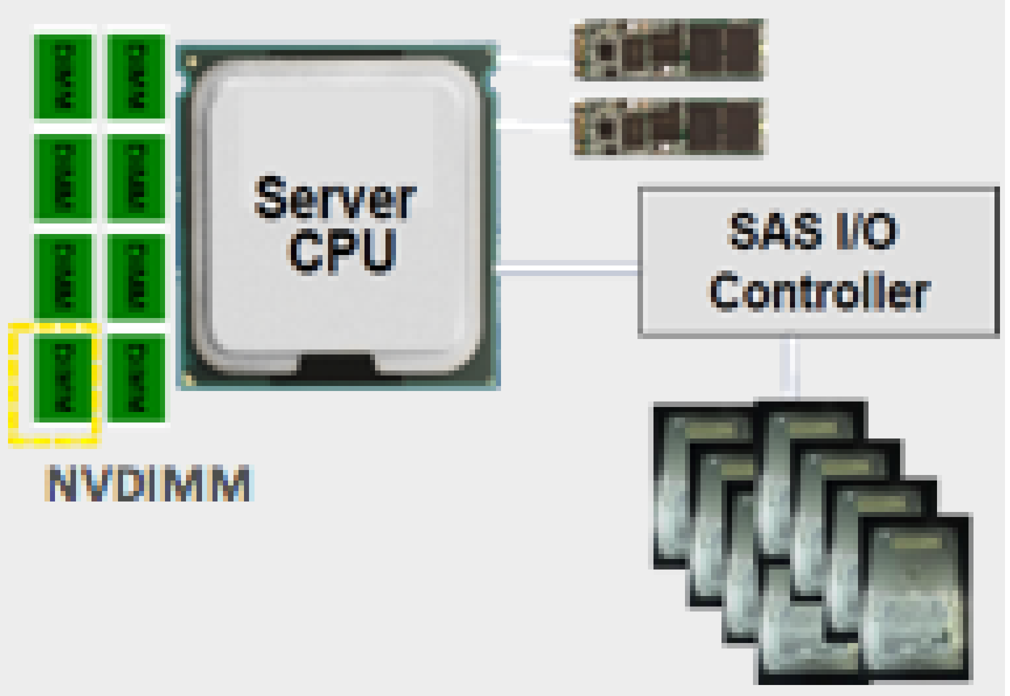
- Extends life of SSD and improves performance for write-intensive apps

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

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

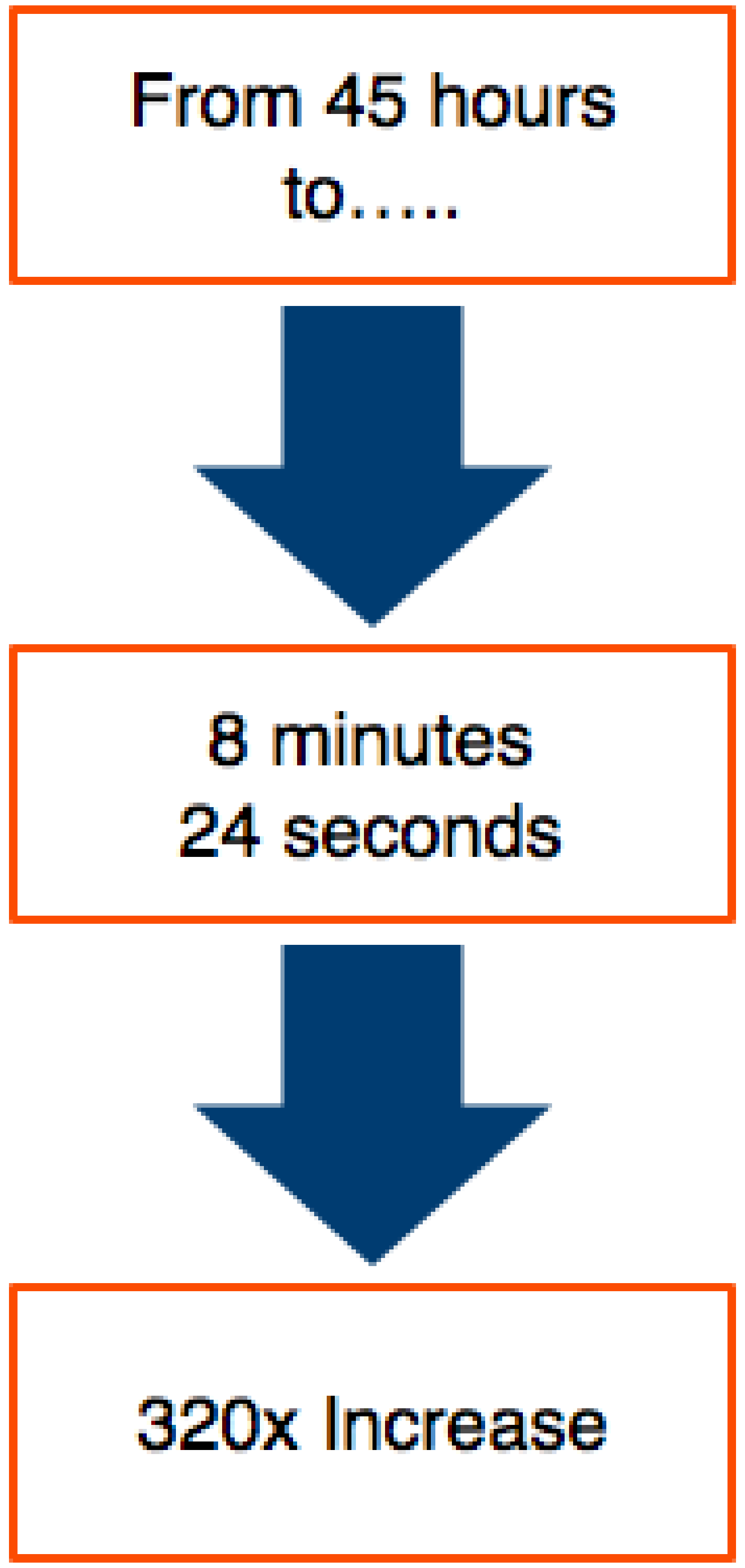
- Forces rethinking of several areas in the software stack:
  - On the OS level:
    - No more need for disk buffer caches
    - File access, still via Posix and/or DMA or some new protocol
  - On the program and framework level:
    - No more need for caching or read-ahead
    - Different ROOT file 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



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);
    }
}
```

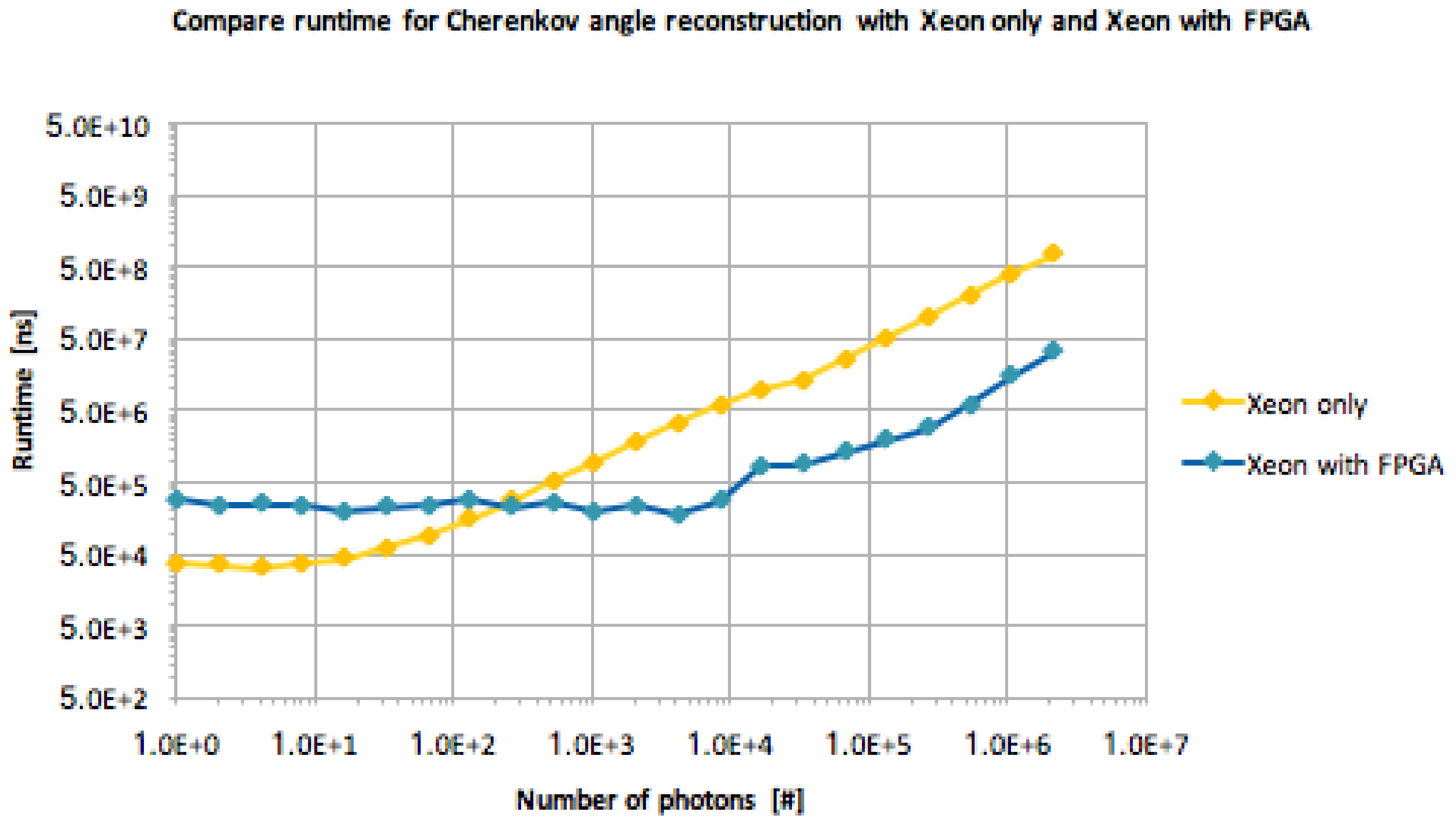
Cilk+ added another 10% speedup

```
#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);
    }
}
```

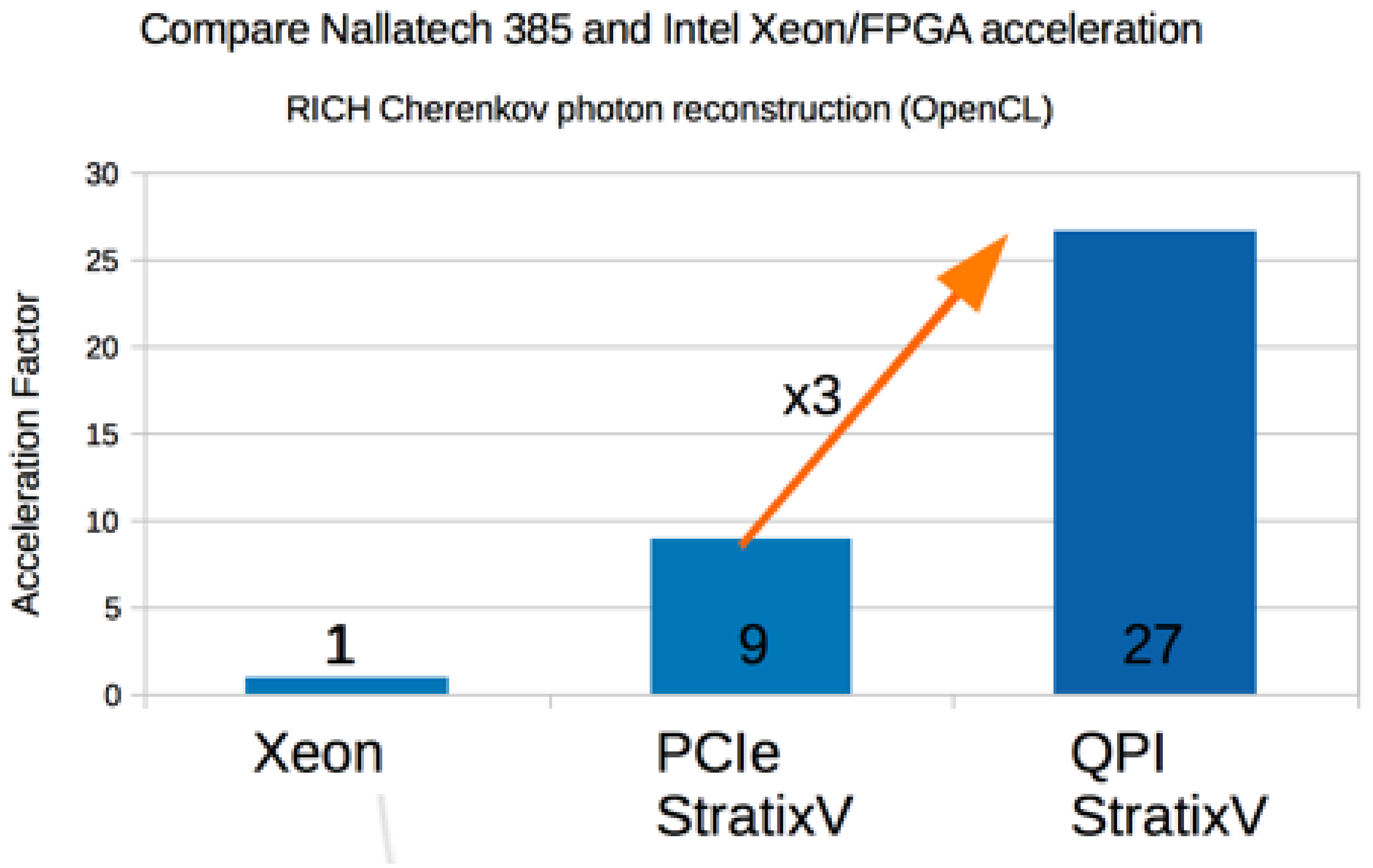
Tightly Integrated Xeon/FPGA Systems

- 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

Conclusions

- CERN openlab, a science–industry partnership that drives R&D and innovation
- A number of very interesting projects underway, with a lot of potential
- Several game changing technologies being researched
- Very interesting times, indeed… no need for doom and gloom

CERN openlab

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