New hardware support in OSIRIS 4.0

R. A. Fonseca$^{1,2}$

$^1$ GoLP/IPFN, Instituto Superior Técnico, Lisboa, Portugal
$^2$ DCTI, ISCTE-Instituto Universitário de Lisboa, Portugal
Outline

• The road to Exascale Systems
  - HPC system evolution
  - Current trends
  - OSIRIS support for top10 systems

• The new Intel Knights Landing architecture
  - Architecture details
  - OSIRIS on the KNL

• OSIRIS on modern CPUs
  - Overview of modern CPUs
  - Compiling and running for these

• OSIRIS on GPUs
The road to Exascale systems
The Road to Exascale Computing

High Performance Computing Power Evolution

Exaflop/s
- 10^{12}
- Sunway TaihuLight
- Tianhe-2
- K Computer
- Titan

Petaflop/s
- 10^{9}
- Cray XT-4
- IBM Blue Gene/L
- IBM RoadRunner
- NEC Earth Simulator
- IBM ASCI White
- Intel ASCI Red/9152

Teraflop/s
- 10^{6}
- Fujitsu Numerical Wind Tunnel
- NEC SX-3/44R
- Cray-2/8
- ETA10-G/8

Gigaflop/s
- 10^{3}
- Cray X-MP/4
- M-13
- CDC Cyber 205
- Burroughs ILLIAC IV

Megaflop/s
- 10^{0}
- CDC 6600
- IBM 7030 Stretch
- IBM 7090
- IBM 709
- UNIVAC 1
- IBM 704
- EDSAC 1

Kiloflop/s
- 10^{-3}

Year
- 1940
- 1950
- 1960
- 1970
- 1980
- 1990
- 2000
- 2010
- 2020

Performance
- \( R_{\text{peak}} \) 125.4 Pflop/s
- \( R_{\text{max}} \) 93.0 Pflop/s

Node Configuration
- 1\times SW26010 manycore processor
- 4\times(64+1) cores @ 1.45 GHz
- 4\times 8 GB DDR3

Total system
- 10 649 600 cores
- 1.31 PB RAM

Sunway Taihulight
- NRCCPC, China
- #1 - TOP500 Jun/16
- 40 960 compute nodes

(data from multiple sources)
The Road to Power Efficient Computing

Energy Requirement Evolution

Energy per operation [mJ]

- UNIVAC 1
- IBM 704
- IBM 7090
- IBM 7030 Stretch
- CDC 6600
- Cray-1
- Cray-2/8
- Fujitsu Numerical Wind Tunnel
- Intel ASCI Red/9152
- IBM ASCI White
- NEC Earth Simulator
- IBM Blue Gene/L
- IBM RoadRunner
- Cray XT5-HE
- K Computer
- Titan
- Tiahne-1A
- Tiahne-2
- Sunway Taihulight

- Energy Requirement Evolution (data from multiple sources)

Sunway Taihulight
- Manycore architecture
- Peak performance 93 PFlop/s
- Total power 15.3 MW
  - 6.07 Gflop/W
  - 165 pJ / flop
The drive towards Exaflop

- Steady progress for over 60 years
  - 138 systems above 1 PFlop/s
- Supported by many computing paradigm evolutions
- Trend indicates Exaflop systems by next decade
- Electric power is one of the limiting factors
  - Target < 20 MW
  - Top system achieves ~ 6 Gflop/W
    - ~ 0.2 GW for 1 Exaflop
    - Factor of 10× improvement still required
  - Best energy efficiency
    - 14 Gflop/W
    - NVIDIA Tesla P100

Multicore systems

- Maintain complex cores and replicate
- 2 systems in the top 10 are based on multicore CPUs
  - 1× Fujitsu SPARK #8 (K-Computer)
  - 1× Intel Xeon E5 #10 (Trinity)

Manicore

- Use many lower power cores
- 5 systems in the top 10
  - #1 (Sunway Taihulight)
  - 2× IBM BlueGene/Q in the top
    - #5 (Sequoia) and #9 (Mira)
    - Seem to be the last of their kind
  - 2× Intel Knights Landing systems
    - #6 (Cori) and #7 (Oakforest-PACS)

Accelerator/co-processor technology

- 3 systems in top 10
  - #3 (Piz Daint) and #4 (Titan) use NVIDIA GPUs
  - #2 (Tianhe-2) uses Intel Knights Corner
### Which top 10 systems are currently supported by OSIRIS?

<table>
<thead>
<tr>
<th>Rank</th>
<th>System</th>
<th>Cores</th>
<th>$R_{\text{max}}$ [TFlop/s]</th>
<th>$R_{\text{peak}}$ [TFlop/s]</th>
<th>Power [kW]</th>
<th>OSIRIS support</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunway TaihuLight, China</td>
<td>10649600</td>
<td>93014.6</td>
<td>125435.9</td>
<td>15371</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Tianhe-2 (MilkyWay-2), China</td>
<td>3120000</td>
<td>33862.7</td>
<td>54902.4</td>
<td>17808</td>
<td>Full (KNC)</td>
</tr>
<tr>
<td>3</td>
<td>Piz Daint, Switzerland</td>
<td>361760</td>
<td>19590.0</td>
<td>25326.3</td>
<td>2272</td>
<td>Full (CUDA)</td>
</tr>
<tr>
<td>4</td>
<td>Titan, United States</td>
<td>560640</td>
<td>17590.0</td>
<td>27112.5</td>
<td>8209</td>
<td>Full (CUDA)</td>
</tr>
<tr>
<td>5</td>
<td>Sequoia, United States</td>
<td>1572864</td>
<td>17173.2</td>
<td>20132.7</td>
<td>7890</td>
<td>Full (QPX)</td>
</tr>
<tr>
<td>6</td>
<td>Cori, United States</td>
<td>622336</td>
<td>14014.7</td>
<td>27880.7</td>
<td>3939</td>
<td>Full (KNL)</td>
</tr>
<tr>
<td>7</td>
<td>Oakforest-PACS, Japan</td>
<td>556104</td>
<td>13554.6</td>
<td>24913.5</td>
<td>2719</td>
<td>Full (KNL)</td>
</tr>
<tr>
<td>8</td>
<td>K computer, Japan</td>
<td>705024</td>
<td>10510.0</td>
<td>11280.4</td>
<td>12660</td>
<td>Standard (Fortran)</td>
</tr>
<tr>
<td>9</td>
<td>Mira, United States</td>
<td>786432</td>
<td>8586.6</td>
<td>10066.3</td>
<td>3945</td>
<td>Full (QPX)</td>
</tr>
<tr>
<td>10</td>
<td>Trinity, United States</td>
<td>301056</td>
<td>8100.9</td>
<td>11078.9</td>
<td>4233</td>
<td>Full (AVX2)</td>
</tr>
</tbody>
</table>
Intel Knights Landing Architecture

Intel Xeon Phi 5110p die
**Intel Knights Landing Architecture**

**NERSC Cori**

- **Cray XC40**
  - 9152 Compute Nodes
  - 3.9 MW
- **Interconnect**
  - Aries Interconnect
- **Node configuration**
  - 1x Intel® Xeon® Phi 7250 @ 1.4GHz (68 cores)
  - 16 GB (MCDRAM) + 96 GB (RAM)

- **Total system**
  - 622 336 cores
  - 0.15 PB (MCDRAM) + 0.88 PB (RAM)
- **Performance**
  - $R_{\text{MAX}} = 14.0$ PFlop/s
  - $R_{\text{PEAK}} = 27.9$ PIFlop/s

**Intel Knights Landing (KNL)**

- **Processing**
  - 68 x86_64 cores @ 1.4 GHz
    - 4 threads / core
    - Scalar unit + 2x 512 bit vector unit
    - 32KB L1-I + 32KB L1-D cache
  - Organized in 36 tiles
    - Connected by 2D Mesh
    - 1 MB L2 cache
    - 2 cores/tile
  - Fully Intel Xeon ISA-compatible
    - Binary compatible with other x86 code
  - No L3 cache (MCDRAM can be configured as an L3 cache)

- **Memory**
  - 16 GB 8-channel 3D MCDRAM
    - 400+ GB/s
  - (Up to) 384 GB 6-channel DDR4
    - 115.2 GB/s
  - 8x2 memory controllers
  - Up to 350 GB/s

- **System Interface**
  - Standalone host processor
  - Max power consumption 215W

---

NERSC Cori Cray XC40 #5 - TOP500 Nov/16

---

Intel Knights Landing Architecture

NERSC Cori Cray XC40

---

Intel Knights Landing (KNL)

- **Processing**
  - 68 x86_64 cores @ 1.4 GHz
    - 4 threads / core
    - Scalar unit + 2x 512 bit vector unit
    - 32KB L1-I + 32KB L1-D cache
  - Organized in 36 tiles
    - Connected by 2D Mesh
    - 1 MB L2 cache
    - 2 cores/tile
  - Fully Intel Xeon ISA-compatible
    - Binary compatible with other x86 code
  - No L3 cache (MCDRAM can be configured as an L3 cache)

- **Memory**
  - 16 GB 8-channel 3D MCDRAM
    - 400+ GB/s
  - (Up to) 384 GB 6-channel DDR4
    - 115.2 GB/s
  - 8x2 memory controllers
  - Up to 350 GB/s

- **System Interface**
  - Standalone host processor
  - Max power consumption 215W

---

NERSC Cori Cray XC40

---

Intel Knights Landing Architecture

NERSC Cori Cray XC40

---

Intel Knights Landing (KNL)

- **Processing**
  - 68 x86_64 cores @ 1.4 GHz
    - 4 threads / core
    - Scalar unit + 2x 512 bit vector unit
    - 32KB L1-I + 32KB L1-D cache
  - Organized in 36 tiles
    - Connected by 2D Mesh
    - 1 MB L2 cache
    - 2 cores/tile
  - Fully Intel Xeon ISA-compatible
    - Binary compatible with other x86 code
  - No L3 cache (MCDRAM can be configured as an L3 cache)

- **Memory**
  - 16 GB 8-channel 3D MCDRAM
    - 400+ GB/s
  - (Up to) 384 GB 6-channel DDR4
    - 115.2 GB/s
  - 8x2 memory controllers
  - Up to 350 GB/s

- **System Interface**
  - Standalone host processor
  - Max power consumption 215W

---

NERSC Cori Cray XC40

---

Intel Knights Landing Architecture

NERSC Cori Cray XC40

---

Intel Knights Landing (KNL)

- **Processing**
  - 68 x86_64 cores @ 1.4 GHz
    - 4 threads / core
    - Scalar unit + 2x 512 bit vector unit
    - 32KB L1-I + 32KB L1-D cache
  - Organized in 36 tiles
    - Connected by 2D Mesh
    - 1 MB L2 cache
    - 2 cores/tile
  - Fully Intel Xeon ISA-compatible
    - Binary compatible with other x86 code
  - No L3 cache (MCDRAM can be configured as an L3 cache)

- **Memory**
  - 16 GB 8-channel 3D MCDRAM
    - 400+ GB/s
  - (Up to) 384 GB 6-channel DDR4
    - 115.2 GB/s
  - 8x2 memory controllers
  - Up to 350 GB/s

- **System Interface**
  - Standalone host processor
  - Max power consumption 215W

---

NERSC Cori Cray XC40

---

Intel Knights Landing Architecture

NERSC Cori Cray XC40

---

Intel Knights Landing (KNL)

- **Processing**
  - 68 x86_64 cores @ 1.4 GHz
    - 4 threads / core
    - Scalar unit + 2x 512 bit vector unit
    - 32KB L1-I + 32KB L1-D cache
  - Organized in 36 tiles
    - Connected by 2D Mesh
    - 1 MB L2 cache
    - 2 cores/tile
  - Fully Intel Xeon ISA-compatible
    - Binary compatible with other x86 code
  - No L3 cache (MCDRAM can be configured as an L3 cache)

- **Memory**
  - 16 GB 8-channel 3D MCDRAM
    - 400+ GB/s
  - (Up to) 384 GB 6-channel DDR4
    - 115.2 GB/s
  - 8x2 memory controllers
  - Up to 350 GB/s

- **System Interface**
  - Standalone host processor
  - Max power consumption 215W

---

NERSC Cori Cray XC40

---

Intel Knights Landing Architecture

NERSC Cori Cray XC40

---

Intel Knights Landing (KNL)

- **Processing**
  - 68 x86_64 cores @ 1.4 GHz
    - 4 threads / core
    - Scalar unit + 2x 512 bit vector unit
    - 32KB L1-I + 32KB L1-D cache
  - Organized in 36 tiles
    - Connected by 2D Mesh
    - 1 MB L2 cache
    - 2 cores/tile
  - Fully Intel Xeon ISA-compatible
    - Binary compatible with other x86 code
  - No L3 cache (MCDRAM can be configured as an L3 cache)

- **Memory**
  - 16 GB 8-channel 3D MCDRAM
    - 400+ GB/s
  - (Up to) 384 GB 6-channel DDR4
    - 115.2 GB/s
  - 8x2 memory controllers
  - Up to 350 GB/s

- **System Interface**
  - Standalone host processor
  - Max power consumption 215W

---

NERSC Cori Cray XC40
### Access Modes

**Cache mode**
- SW-Transparent
- Covers whole DDR range

**Flat mode**
- MCDRAM as regular memory
- SW-Managed
- Same address space

**Also a hybrid Mode**
- Use MCDRAM as part memory part cache
- 25% or 50% as cache

### Clustering Modes

**All2All**
- No affinity between tile, directory and memory
- Most general, usually lower perf. than other modes

**Quadrant**
- Chip divided into 4 virtual quadrants
- Affinity between directory and memory, Lower latency and better bandwidth than All2All, SW-transparent

**Sub-NUMA clustering**
- Looks like 4-socket Xeon
- Affinity between tile, directory and memory, Lowest latency, requires NUMA optimize to get benefit

**Memory access / cluster modes must be configured at boot time**

### KNL Mesh Interconnect
Logical view of a KNL node

Intel KNL systems

- The KNL node is simply a 68 core (4 threads per core) shared memory computer
  - Each core has dual 512 but VPUs

- This can be viewed as small computer cluster with \textbf{N} SMP nodes with \textbf{M} cores per node
  - Use a distributed memory algorithm (\texttt{MPI}) for parallelizing across nodes
  - Use a shared memory algorithm (\texttt{OpenMP}) for parallelizing inside each node
  - The exact same code used in "standard" HPC systems can run on the KNL nodes

- The vector units play a key role in system performance
  - Vectorization for this architecture must be considered
Parallelization for a KNL node

- HPC systems present a hierarchy of distributed / shared memory parallelism

- Top level is a network of computing nodes
  - Nodes exchange information through network messages

- Each computing node is a set of CPUs / cores
  - Memory can be shared inside the node

- Parallelization of PIC algorithm
  - Spatial decomposition across nodes: each node handles a specific region of simulation space
  - Split particles over cores inside node
  - Same as for standard CPU systems

**Distributed Memory (MPI)**

**Shared Memory (OpenMP)**
Vectorization of the PIC algorithm for KNL

- The code spends ~80-90% time advancing particles and depositing current
  - Focus vectorization effort here

- Previous strategy developed for vectorization works well
  - Process \( n_v \) (16) particles at a time for most of the algorithm

- For current deposition vectorize by current line to avoid serialization
  - Critical because of large vector width
  - Use a vector transpose operation

**Particle Push**
- Load \( n_v \) particles into Vector Unit
- Interpolate Fields
- Push Particles
- Split Path / Create virtual particles
- Store Results

**Current Deposition**
- Load \( n_v \) virtual part. into Vector Unit
- Calculate Currents
- Transpose current to get 1 vector per current line
- Loop over vectors and accumulate current

**Vectorized by particle**
- Vectors contain same quantity for all particles

**Transpose vectors**
- Done through vector operations

**Vectorized by current line**
- Vectors contain 1 line of current for 1 particle

- The code spends ~80-90% time advancing particles and depositing current
  - Focus vectorization effort here

- Previous strategy developed for vectorization works well
  - Process \( n_v \) (16) particles at a time for most of the algorithm

- For current deposition vectorize by current line to avoid serialization
  - Critical because of large vector width
  - Use a vector transpose operation
Warm plasma tests
- 2D Problem
- 3400 $\times$ 1600 cells
- $340 \times 160 (c/\omega_p)^2$ box
- 25 particles / cell
- 136M particles total

Configuration
- AVX-512 manual vectorization using intrinsics
- No OpenMP parallelism
- 272 MPI processes
  - 4 processes/core
- MCDRAM configuration
  - Flat / SNC4
  - Full problem in MCDRAM

Performance
- Measured as number of particle pushes per second
- Measurement includes all the simulation loop, including the field solver and communications

![2D Code performance graph]

Performance [MPart/s]
- Linear: 1,369
- Quadratic: 925
- Cubic: 628

Interpolation Level
- Linear
- Quadratic
- Cubic

Test system software
- CentOS 7.3
- Intel(R) Parallel Studio XE 2017 Update 3 for Linux
- Intel(R) MPI Library for Linux* OS, Version 2017 Update 2

Compiler Flags
- `% mpiifort -xmic-avx512 -O3 -no-prec-div -fp-model fast=2 -fma -align array64byte -finline-functions`
- `% mpiicc -xmic-avx512 -O3 -no-prec-div -fp-model fast=2 -ansi-alias -fma -align -finline-functions`
Processes per core / Hyperthreading

**KNL cores support 4 threads per core**
- Hide memory latency
- Saturate VPUs

**Test configuration**
- Everything the same as reference configuration except
- 68 - 272 MPI processes
  - 4 processes/core

**Impact on Performance**
- 38,6% drop on performance for linear interpolation from 4 to 1 processes per core
- Slightly smaller drop for cubic interpolation because of increased computational intensity
- Use of 4 threads per core recommended
OSIRIS supports hybrid MPI/OpenMP parallelism
- Improve load imbalance
- Allow scalability for higher core counts
  - Minimal size per core for MPI parallelism
- OpenMP algorithm not as efficient as MPI for uniform plasmas

Test configuration
- Everything the same as reference configuration except...
- Parallel partition \(n_p\) MPI processes \(\times n_t\) OpenMP threads
- \(n_t = 1, 2, 4, 8\) and 16 threads
- \(n_p \times n_t = 272\) for all tests
  - 4 threads per core always

Impact on Performance
- As expected, performance drops with increasing number of threads
- Negligible drop in performance (<~ 5%) for up to 4 threads
- Even at 16 threads performance drop is acceptable, especially for higher interpolation levels
**Explicit (intrinsics) vs. Automatic (compiler) Vectorization**

**KNL cores have 2× 512bit vector units**
- Critical for performance
- Compiler can automatically vectorize the code to use these units
- Can be programmed explicitly using AVX-512 intrinsics

**Test configuration**
- Everything the same as reference configuration except...
- Particle push is now done using the reference fortran pusher using automatic vectorization

**Impact on Performance**
- Explicit vectorization using AVX-512 intrinsics achieves speedups from 3.5× to 5.3×
- The reference fortran code has not been modified (i.e. using pragmas to guide vectorization), may have room for improvement
Memory access/clustering mode

**KNL MCDRAM memory supports many configurations**
- Access mode can be flat / cache
- Memory interconnect can be set to SNC4, Quadrant and All2All

**Impact on Performance**
- Running the code completely in the MCDRAM yields best results
- Cache mode is extremely efficient, performance drop is below 2%
- Running the code strictly on DDR memory gives a 2.17× slowdown in linear interpolation
  - On higher interpolation levels impact much lower, for cubic only a 1.36× slowdown
- SCN4 faster than Quadrant and All2All, difference between Quadrant and All2All negligible
  - Again impact on lower interpolation levels more significant
**Knights Landing vs. Knights Corner**

**KNL is the 2nd generation Intel MIC architecture**
- First generation was the Knights Corner architecture
- Available only as coprocessor boards
- KNC board configuration
  - 60 x86_64 cores @ 1.053 GHz
  - 1x 512bit VPU/core
  - 8 GB GDDR5 RAM
  - Peak FP ~ 2 TFlop/s (SP)
- KNL main differences
  - More cores / higher clock speed
  - Twice the VPU units / core
  - 16 GB MCDRAM
  - Peak FP ~ 6 TFlop/s (SP)

**Programming for KNL vs. KNC**
- KNC intrinsics almost identical to AVX512 with a few exceptions, small changes required to vector code
- KNL has additional instructions for unaligned memory access
- Also additional AVX512 instructions (e.g. conflict detection), not explored yet.

**KNL vs. KNC performance**
- Avg. speedup was 2.2×
- Floating point efficiency lower on KNL
- Room for improvement on KNL code

**KNL vs. KNC performance**

<table>
<thead>
<tr>
<th>Interpolation Level</th>
<th>Speedup KNL/KNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>2.26 ×</td>
</tr>
<tr>
<td>Quadratic</td>
<td>2.18 ×</td>
</tr>
<tr>
<td>Cubic</td>
<td>2.13 ×</td>
</tr>
</tbody>
</table>
Using OSIRIS on QPX / AVX* systems

Intel Xeon Phi 7290 die
## SIMD support on modern CPUS

### SIMD extension support

<table>
<thead>
<tr>
<th>Name</th>
<th>SSE2</th>
<th>AVX</th>
<th>AVX2</th>
<th>AVX512</th>
<th>KNC ²</th>
<th>QPX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>x86</td>
<td>x86</td>
<td>x86</td>
<td>x86</td>
<td>x86</td>
<td></td>
</tr>
<tr>
<td>Width [bits]</td>
<td>128</td>
<td>256</td>
<td>256</td>
<td>512</td>
<td>512</td>
<td>256</td>
</tr>
<tr>
<td>Fused multiply-add operations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Vector Gather / Scatter instructions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Float vector width</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>n/a</td>
</tr>
<tr>
<td>Double vector width</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Integer vector width</td>
<td>4</td>
<td>4+4</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>n/a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System</th>
<th>Jaguar</th>
<th>Supermuc</th>
<th>Trinity</th>
<th>Marenostrum 4</th>
<th>Tianhe-2</th>
<th>Sequoia</th>
</tr>
</thead>
</table>

*AVX-512 extensions are available on both Intel Knights Landing and Xeon Platinum CPUs*

*KNC systems are not binary compatible with other x86 systems*

---

### Modern CPUS include SIMD vector unit(s)

- Vector registers of $n_V$ values (int/float/double)
- Instructions act on vector registers
- Same operation on $n_V$ values simultaneously
- Most also include support for fused multiply-add operations
  - $a \times b + c$ in a single op

### Programming SIMD units

- Let the compiler do it
  - Developer may give compiler hints to help automatic vectorization of code
- Use intrinsics
  - Special operations that translate to vector operations directly

### OSIRIS includes support for all the main SIMD units

- Explicit vectorization of the particle advance/deposit routines
- Automatic vectorization of the field solver routines
Using OSIRIS with vector (SIMD) hardware

@ Compile time

- Make sure you have enabled the correct SIMD flag on the configure file

```
# SIMD
SIMD = AVX2
```

- Available options are
  - **QPX** - PowerPC A2 CPUs (BlueGene/Q systems)
  - **AVX2** - x86 CPUs supporting AVX2 extensions, such as Intel Kaby Lake or AMD Ryzen
  - **AVX512** - x86 CPUs supporting AVX512 extensions, such as Intel Knights Landing or Intel Xeon Platinum

- Legacy options
  - **SSE** - Really old x86 systems
  - **AVX** - Not so old x86 systems that do not yet support AVX2 (e.g. my previous laptop)

- **SSE/AVX* options are forward compatible**
  - e.g. AVX code will run on AVX512 cpus

@ Run time

- **OSIRIS will always compile the standard Fortran advance/deposit also**
- **Users can choose between them using the push_type parameter in the species section**

```fortran
species
{
    name = "positrons",
    num_par_x(1:2) = 4, 4,
    rqm=+1.0,
    ! push_type = "standard",
    push_type = "simd",
}
```

- **When SIMD support was chosen at the compile stage “simd” will be the default**
  - Users can choose the standard pusher using the “standard” option
  - *Can also be used to choose the radiation cooling pusher (“radcool”)*
• Simulation setup
  - Collision of two neutral electron/positron clouds
  - Physics dominated by the Weibel instability
  - 2D simulation in the perpendicular plane

• Parameters
  - 8192 × 8192 cells
  - 4 species, 4.2×10⁹ particles total
  - γ v_{fi} = ± 0.6 c
  - γ v_{th} = 0.1 c
  - SNC4 / Cache
  - 50 GB ~ 50% system memory (DDR) used

• Very good performance on a single KNL node
Overview

Harvard Mark I - 1944
Rear view of Computing Section
Overview

• **Outstanding progress in computational power since 1950s**
  - Present systems can reach performances of 0.1 EFlop/s
  - Energy cost for calculations has gone down by 14 orders of magnitude
  - Continuous evolution of architectures and computing paradigms

• **OSIRIS is ready to take advantage of the most advanced computing resources in the world**
  - Only 1 system in the top-10 is not supported
  - 8 systems in the top-10 are fully supported, with computing kernels developed specifically to take advantage of advanced hardware features

• **The Intel Knights Landing Architecture is a promising architecture for PIC simulations**
  - Existing codes and algorithms can be efficiently ported to this architecture
  - Good support for MPI and OpenMP parallelism
  - Explicit vectorization critical for good performance
  - New MCDRAM memory has high impact on performance at lower interpolation orders
General Purpose Graphical Processing Units