Geant4
- General Status Updates and Perspectives

Makoto Asai (SLAC)

May 28th, 2014

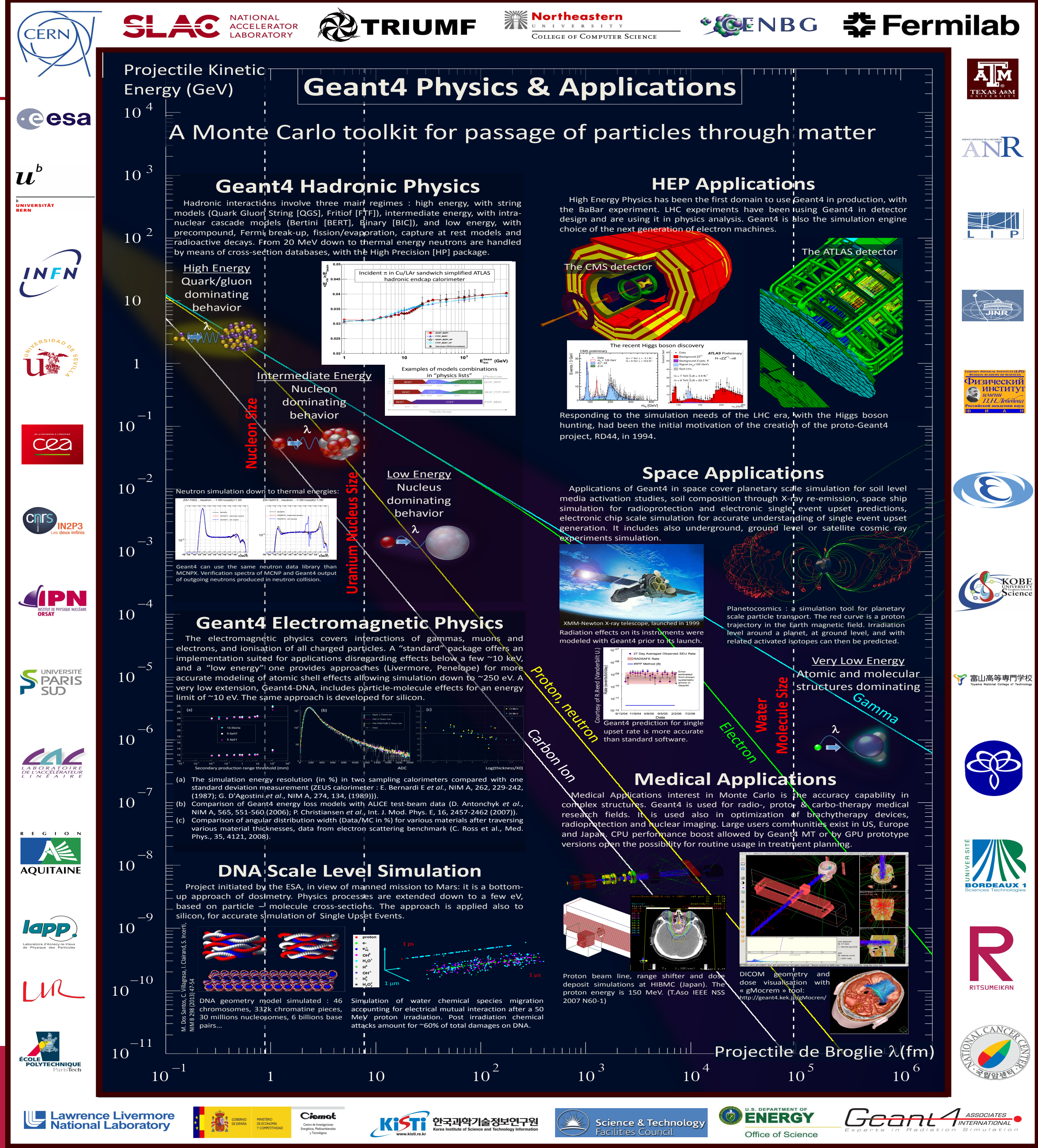
Geant4 Space Users Workshop

NATIONAL ACCELERATOR LABORATORY

U.S. DEPARTMENT OF ENERGY
Office of Science

Contents

- General updates
- Highlights of version 10.0
- Geant4 – the future



Geant4 – Its history

- Dec '94 - Project start
- Apr '97 - First alpha release
- Jul '98 - First beta release
- Dec '98 - First Geant4 public release - version 1.0
- …
- Dec 2nd, '11 – Geant4 version 9.5 release
  - Oct 22nd, '12 - Geant4 9.5-patch02 release
- Nov 30th, '12 – Geant4 version 9.6 release
  - Mar 20th, '14 - Geant4 9.6-patch03 release ← Retroactive patch release
- Dec 6th, '13 – Geant4 version 10.0 release
  - Feb 28th, '14 - Geant4 10.0-patch01 release ← Current version
  - Coming very soon - Geant4 10.0-patch02 release

- We currently provide one public release every year.
  - Beta releases are also available.
  - Release announcements on Collaboration Web pages and through the announcement mailing list

Geant4 version 10 series

- The release in 2013 was a major release.
  - Geant4 version 10 – release date : Dec. 6, 2013
- The highlight is its multi-threading capability.
  - A few interfaces need to be changed due to multi-threading
- It offers two build options.
  - Multi-threaded mode (including single thread)
  - Sequential mode
    - In case a user depends on thread-unsafe external libraries, he may install Geant4 in sequential mode. Almost zero migration cost for sequential v10.

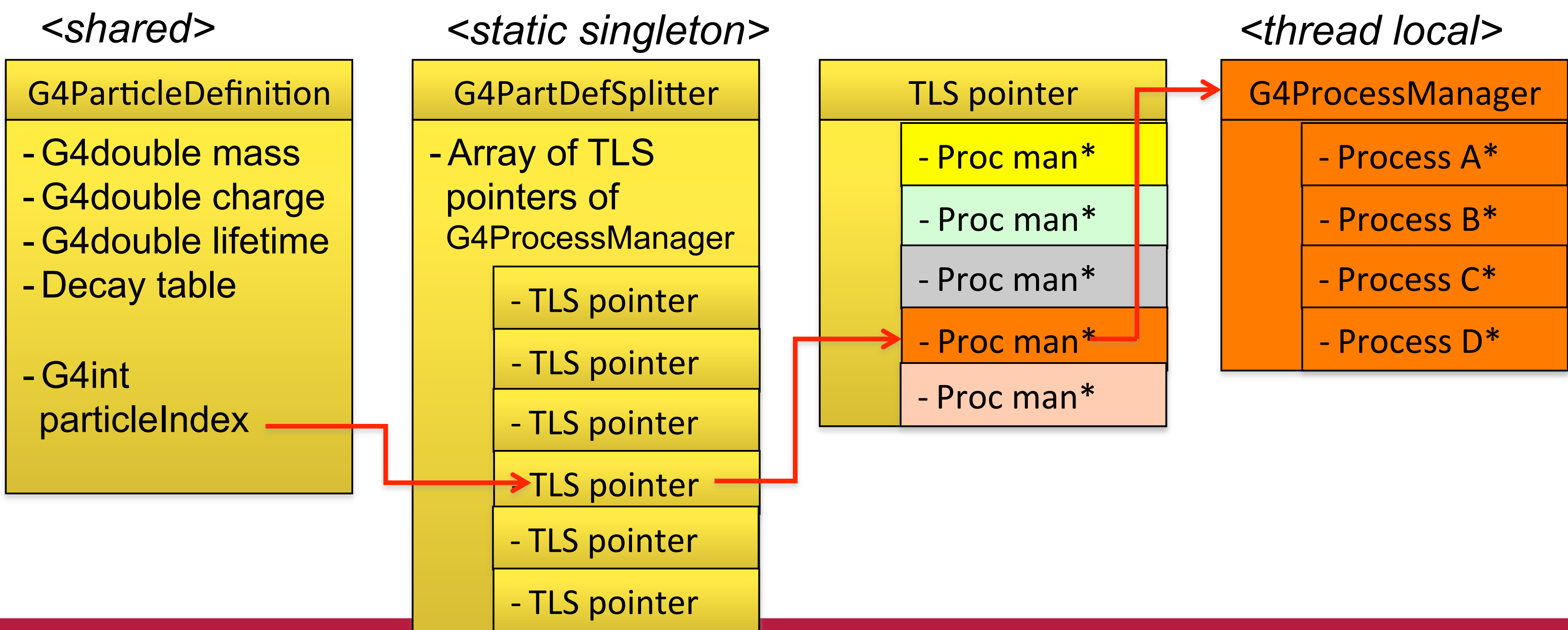
G4MT prototype-9.4 (2011)	G4MT prototype-9.5 (2012)	G4 10.0.beta (June 2013)	G4 10.0 (Dec. 2013)	G4 10 series (2014~)
• Proof of principle • Identify objects to be shared • First testing	• MT code integrated into G4	• API re-design • Example migration • Further testing • First optimizations	• Production ready • Public release	• Further refinements

Geant4 multi-threading : event-level parallelism

- This choice minimizes the changes in user-code
  - Maintain API changes at minimum
- All Geant4 code has been made thread-safe.
  - Thread-safety implemented via Thread Local Storage
- Most memory-consuming parts of the code (geometry, physics tables) are shared over threads.
  - "Split-class" mechanism: reduce memory consumption
    - Read-only part of most memory consuming classes are shared
    - Enabling threads to write to thread-local part
- Particular attention to create "lock-free" code: linearity (w.r.t. #threads) is the metrics we are concentrating on for the v10.0 release.

Split class – case of particle definition

- In Geant4, each particle type has its own dedicated object of G4ParticleDefinition class.
  - Static quantities : mass, charge, life time, decay channels, etc.,
    - To be shared by all threads.
  - Dedicated object of G4ProcessManager : list of physics processes this particular kind of particle undertakes.
    - Physics process object must be thread-local.



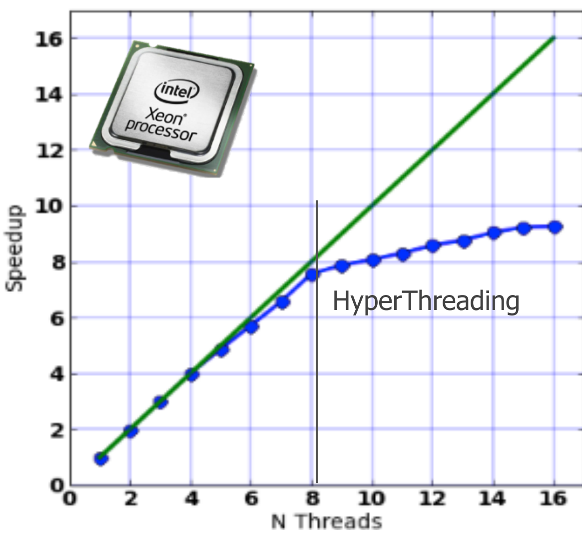
Geant4 multi-threading : event-level parallelism

- This choice minimizes the changes in user-code
  - Maintain API changes at minimum
- All Geant4 code has been made thread-safe.
  - Thread-safety implemented via Thread Local Storage
- Most memory-consuming parts of the code (geometry, physics tables) are shared over threads.
  - "Split-class" mechanism: reduce memory consumption
    - Read-only part of most memory consuming classes are shared
    - Enabling threads to write to thread-local part
- Particular attention to create "lock-free" code: linearity (w.r.t. #threads) is the metrics we are concentrating on for the v10.0 release.
- Initial performance penalties observed in early prototypes have already been addressed.
- Testing on both x86_64 and MIC architectures
- Use of POSIX standards
  - Allowing for integration with user-preferred parallelization frameworks (e.g. MPI, TBB, etc.)

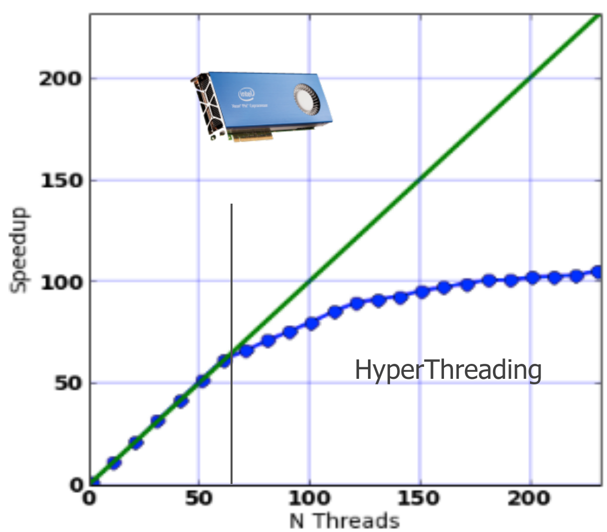
Performance on different architectures

- Current beta release has already shown good scalability on a number of different architectures: Intel Xeon servers, Intel Xeon Phi co-processors and low-power ARM processors.
  - On Intel architectures, it has shown performance improvements not only up to the number of physical cores but in hyper-thread mode as well.

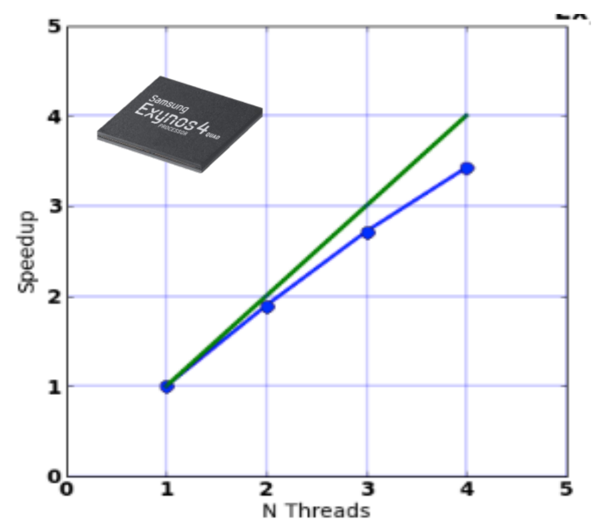
Intel Xeon L5520 @ 2.27GHz



Intel Xeon Phi 7120P @ 1.238GHz

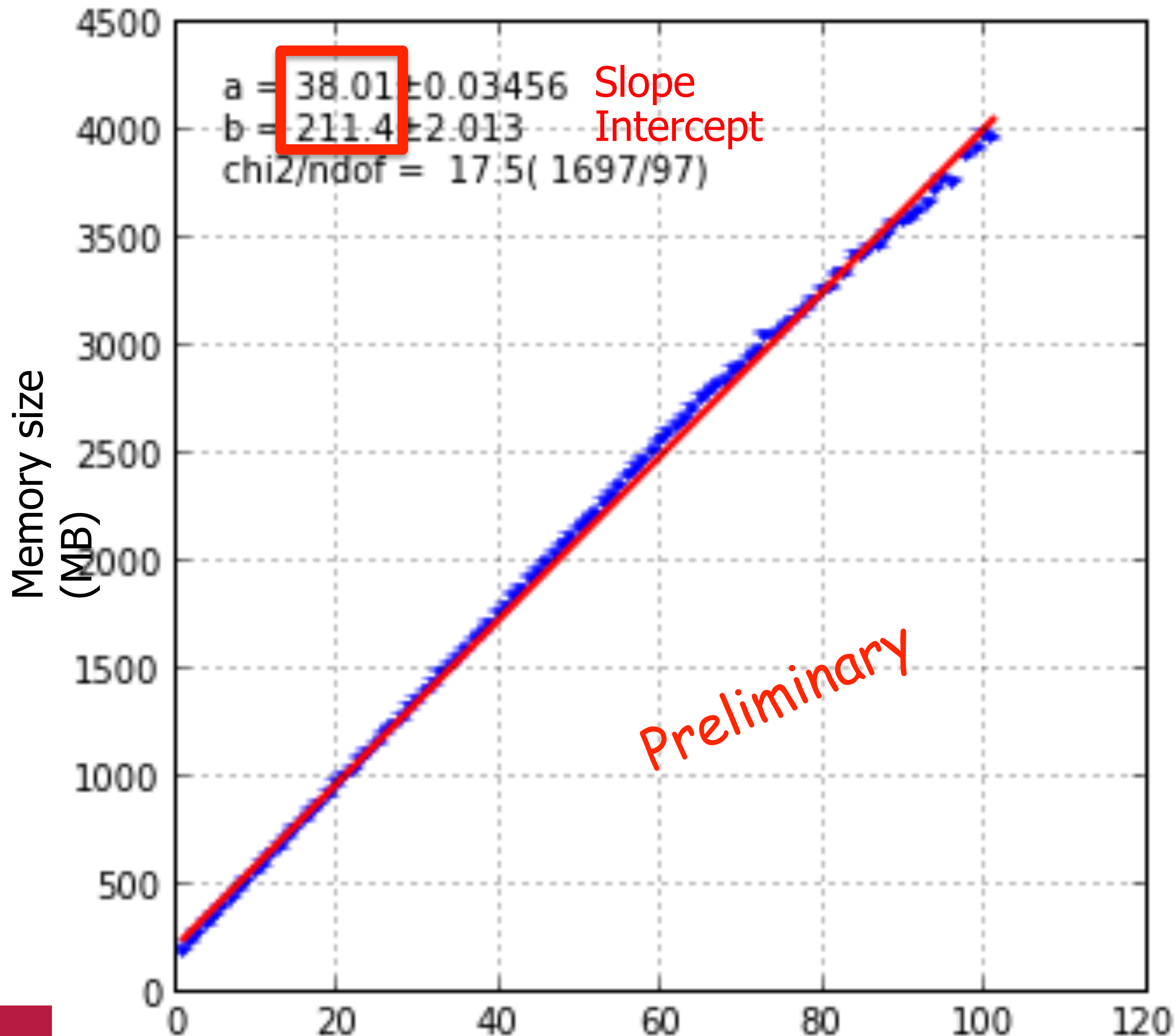


Exynos 4412 Quad-Core @ 1.7 GHz



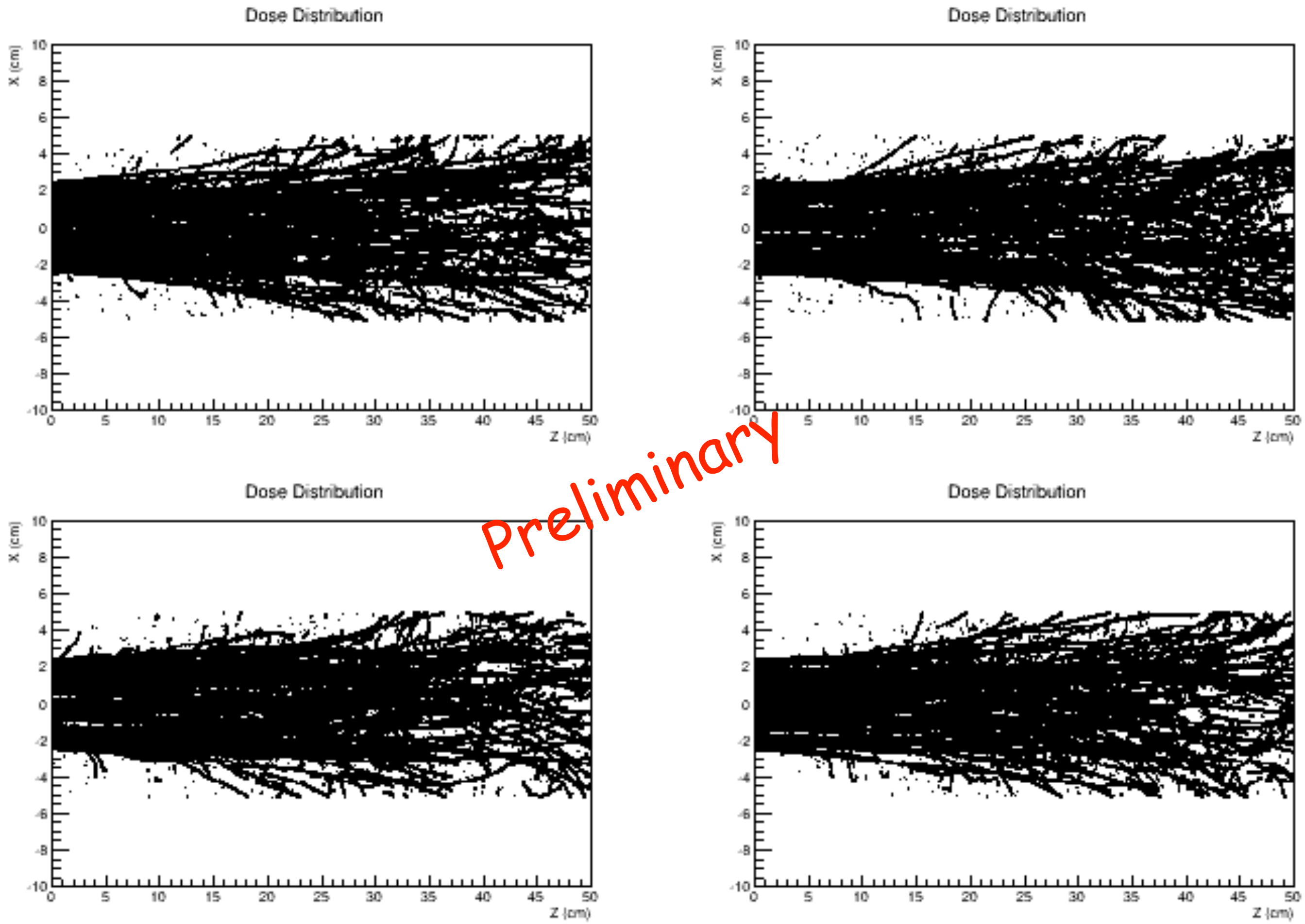
Memory consumption

- Geant4 compiled for MIC architecture
- Full CMS detector without sensitive detectors, hits or trajectories
- No optimization yet
- ~40MB /thread
- Works in progress to reduce the memory consumption per thread.
    - For example eliminating big static arrays in physics processes



MPI + multi-threading

- Geant4 version 10 works with MPI.
  - Many nodes of many cores



- 4 MPI processes with 2 cores each
- Each MPI process owns histogram
- Threads merge dose calculation in shared histogram

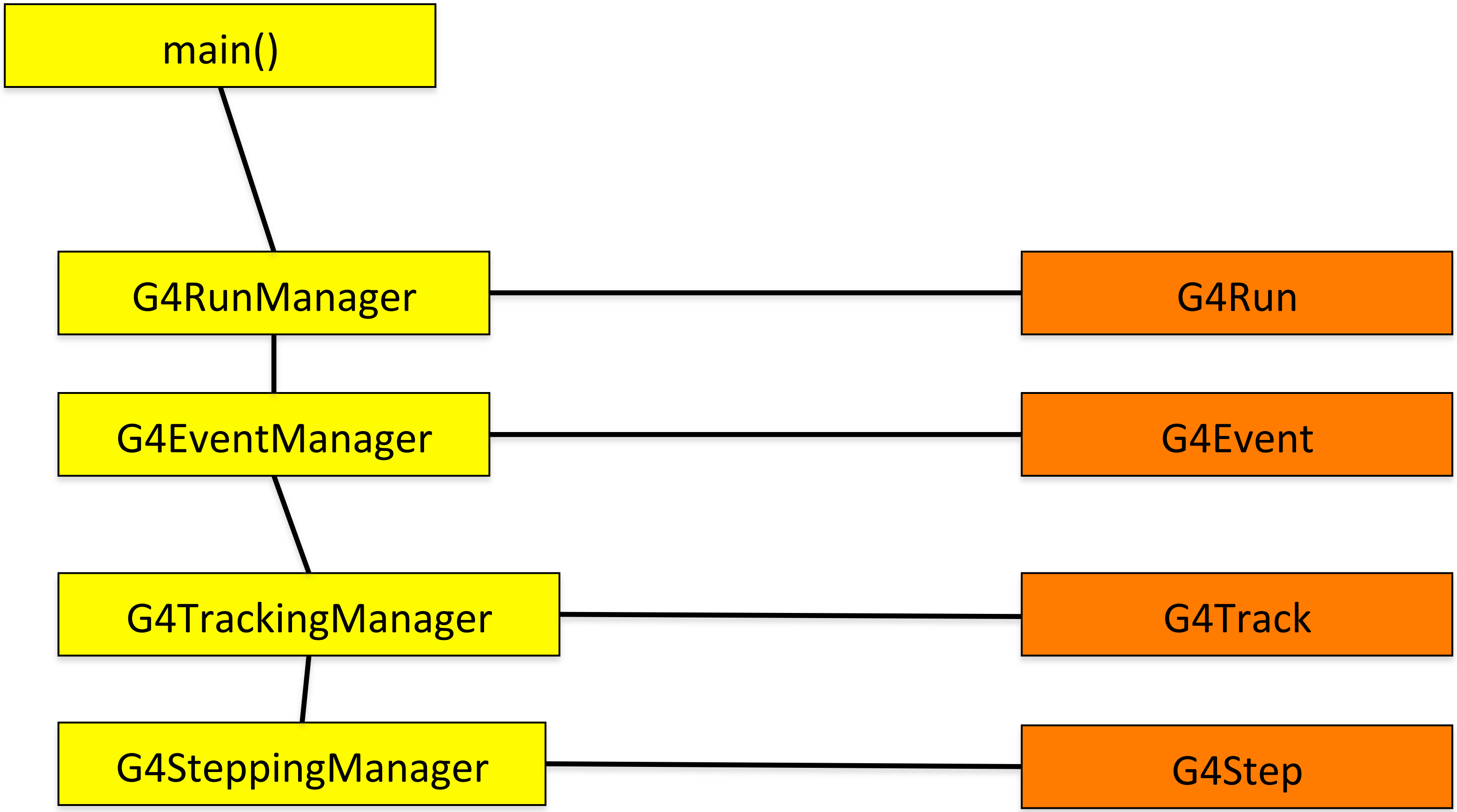
Preliminary studies on TBB

- Intel Threading Building Block is a library for task-based multi-threading code. Some LHC experiments show their interest in the use of TBB in their frameworks.
- We have verified that the G4MT can be used in a TBB-based application where TBB-tasks are responsible for simulating events.
  - We didn't need to modify any concrete G4MT class/method to adapt to TBB.
- We provide an example in version 10 release to demonstrate the way of integrating TBB and G4MT.
- We keep investigating where/how to reduce memory use.
  - We will keep communicating with our users to polish our top-level interfaces.

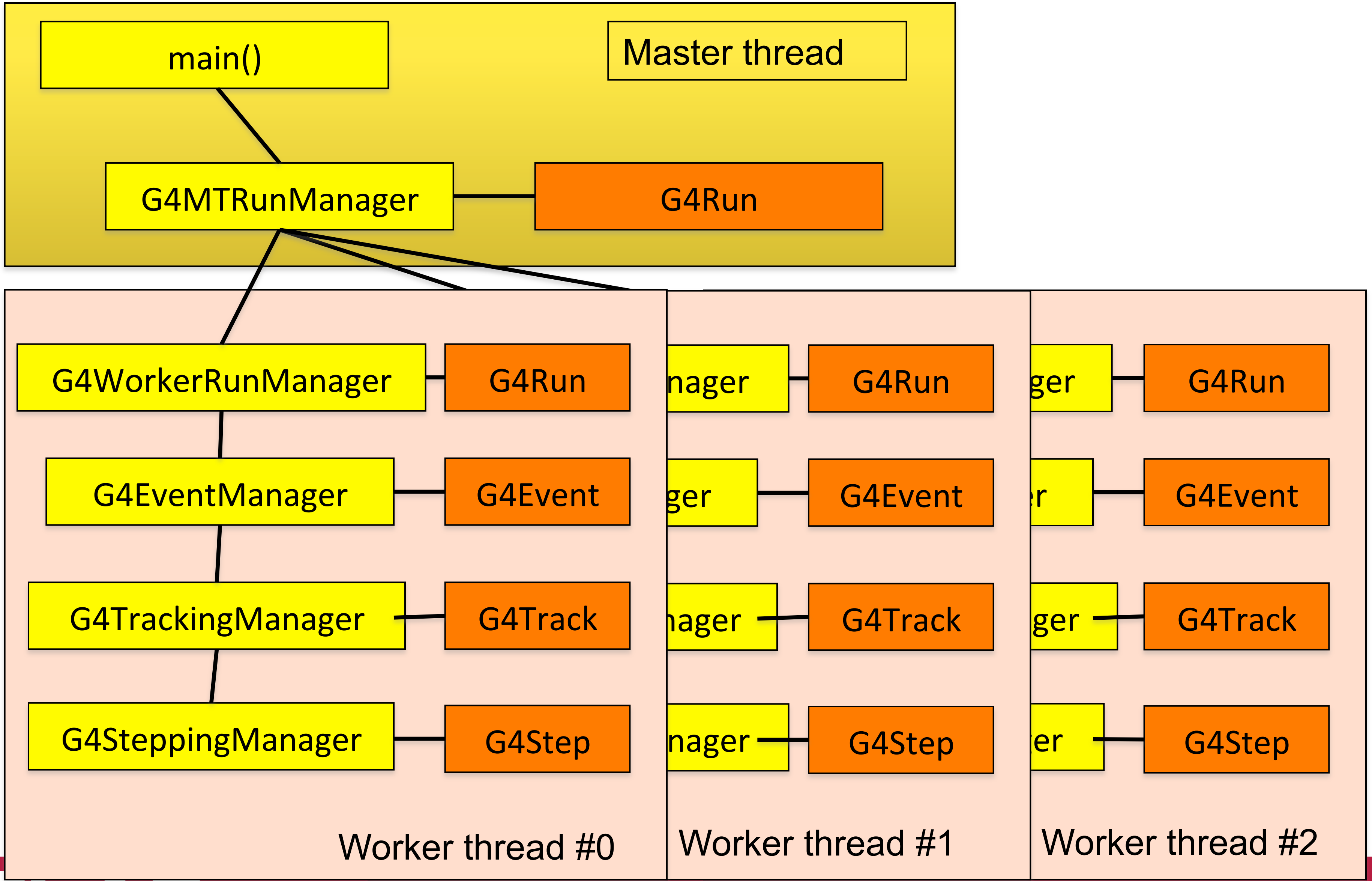
User's code migration

- Migration of user's code to Geant4 version 10 should be fairly easy and straightforward.
  - Migration guide is available.
- G4MTRunManager collects run objects from worker threads and "reduces".
  - Don't accumulate values in user-action classes, but use run class.
    - If you are accumulating quantities in your tracking action or stepping action in your current application, you should note that these action classes will be thread-local.
  - Scores of built-in command-based scorers are automatically reduced.
- Every file I/O for local thread is a challenge
  - Input : primary events : examples are offered in the migration guide.
  - Output : event-by-event hits, trajectories
- Histograms
  - ROOT is thread-unsafe. Geant4 analysis tool (ROOT-bound) is thread-safe.
- It is always a good idea to clearly identify which class objects are thread-local.

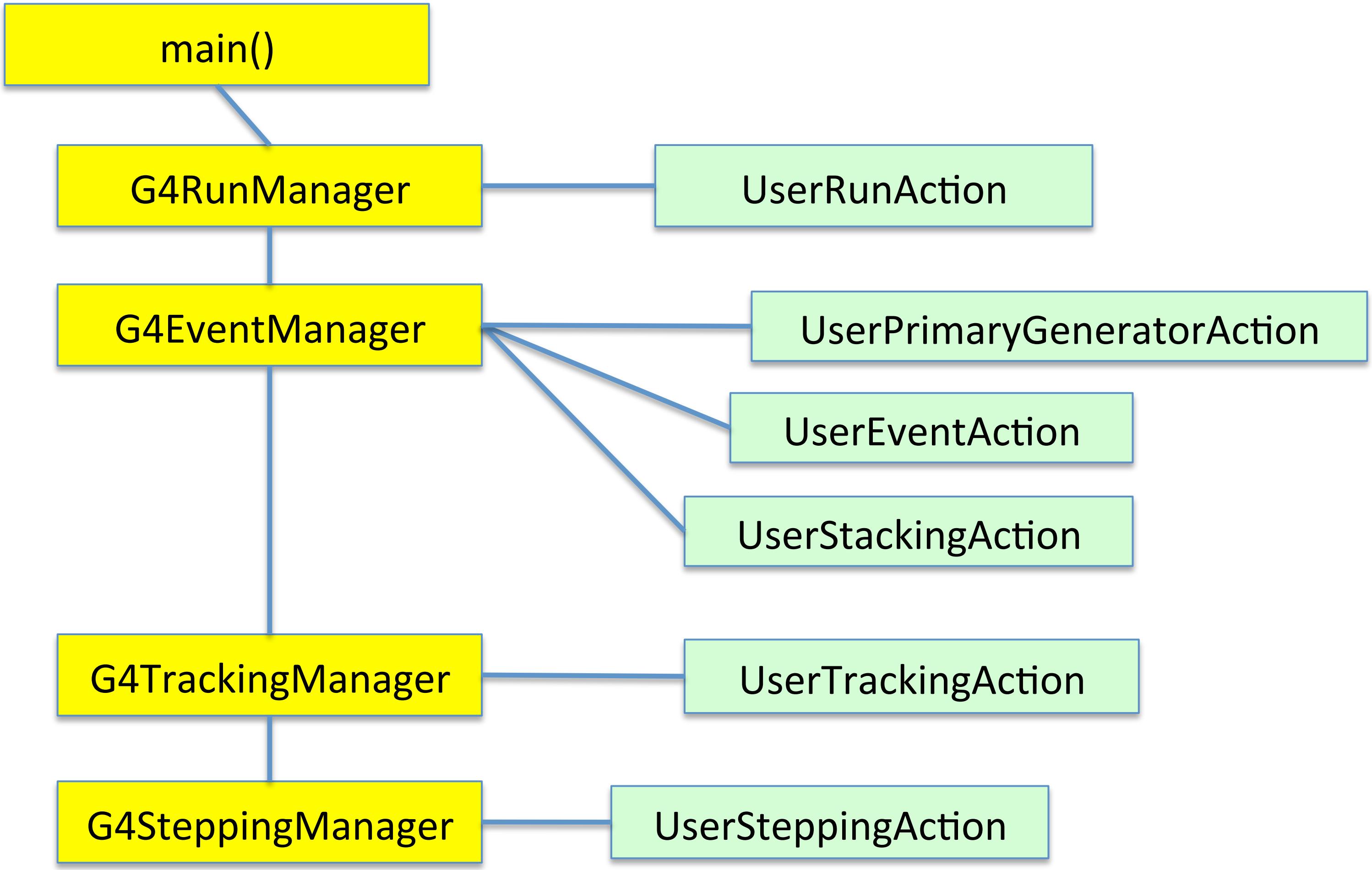
Sequential mode



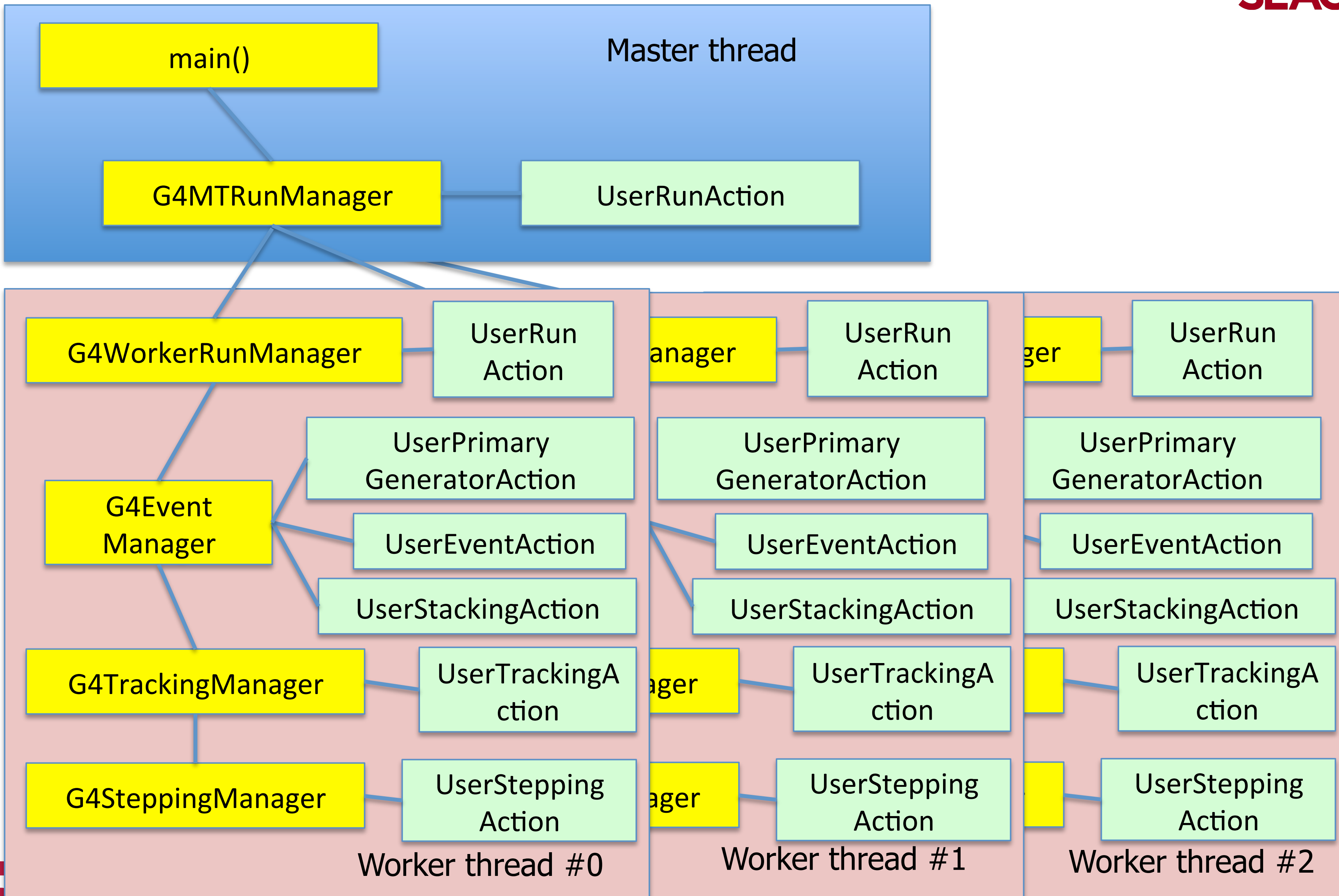
Multi-threaded mode



Sequential mode

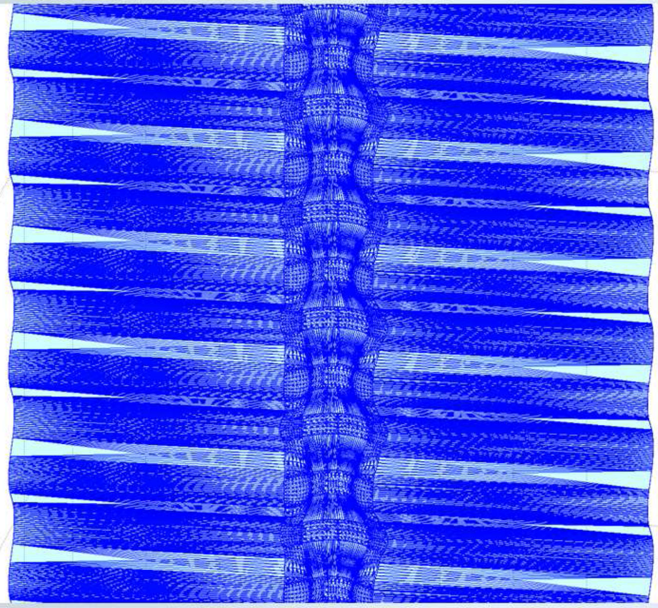


Multi-threaded mode



Geometry updates – New solid library

- An important effort was begun in the last couple of years to write a new solid library, reviewing at the algorithmic level most of the primitives and provides an enhanced, optimized and well-tested implementation to be shared among software packages.
- In most cases considerable performance improvement was achieved.
  - For example, the time required to compute intersections with the tessellated solid was dramatically reduced with the adoption of spatial partitioning for composing facets into a 3D grid of voxels.
- Such techniques allow speedup factors of a few thousand for relatively complex structures having of order 100k to millions of facets, which is typical for geometry descriptions imported from CAD drawings.
  - Consequently, it is now possible to use tessellated geometries for tuning the precision in simulation by increasing the mesh resolution, something that was not possible before.

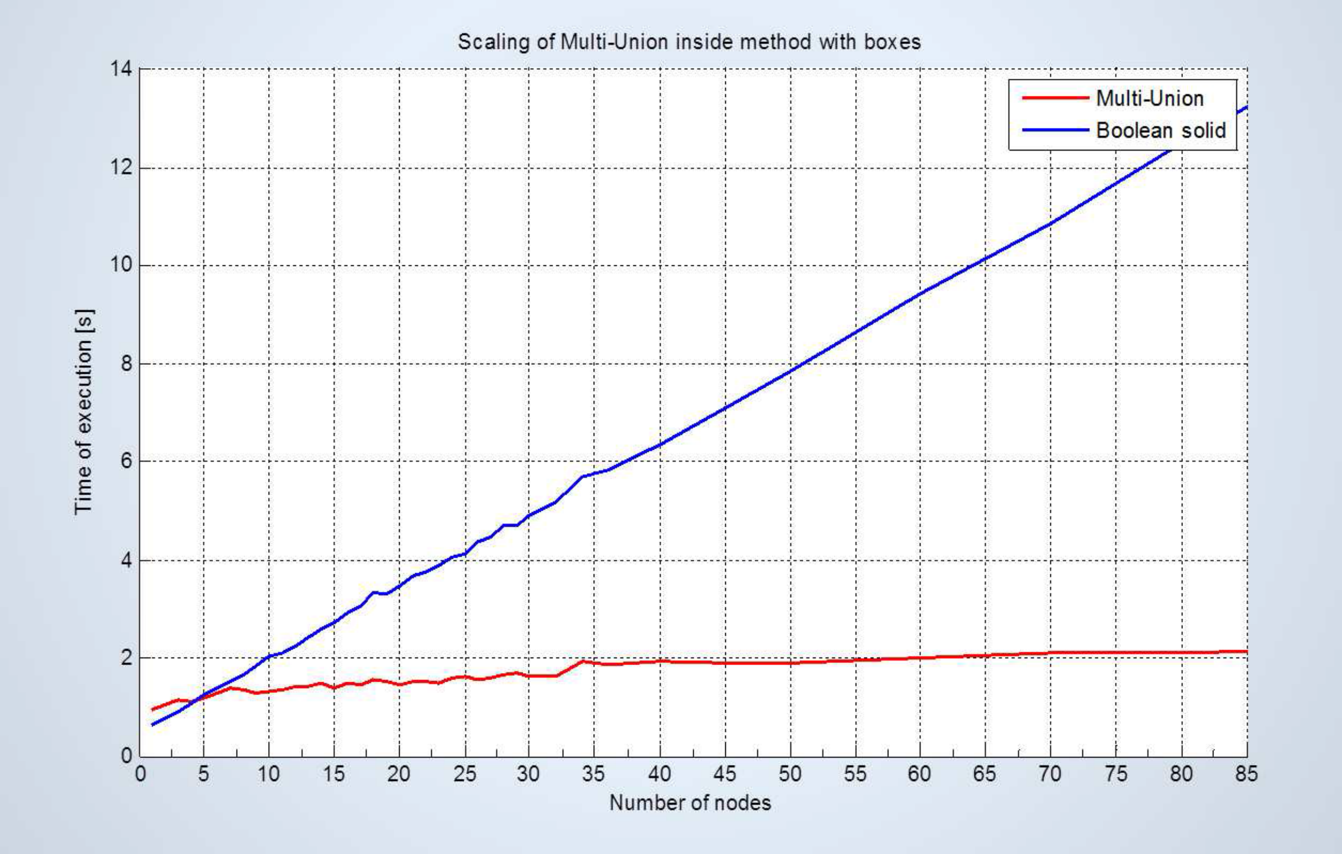


Method	Speedup
Inside	2423x
DistanceToIn	1334x
DistanceToOut	1976x

Information	Value
Number of facets	164.149
Number of voxels	100.000
Memory saved compared with original Geant4	22% (51MB)

Geometry updates – New “multi-union” solid

- In addition to a full set of highly optimized primitives and a tessellated solid, the library includes a new "multi-union” structure implementing a composite set of many solids to be placed in 3D space.
- This differs from the simple technique based on Boolean unions, with the aim of providing excellent scalability on the number of constituent solids.
- The multi-union adopts a similar voxelization technique to partition 3D space, allowing dramatically improved speed and scalability over the original implementation based on Boolean unions.



Highlights of other new features in Geant4 version 10 (1)

- EM physics
  - Studies and improvements in PAI models
    - Validation studies done : G4PAIModel and G4PAIPhotonModel (based on photoabsorption theory) provide better agreement with data than G4UniversalFluctuations (parameterization, default model)
    - Improved angular distribution of delta-electrons
  - Muon/hadron radiative processes
    - Max energy dependent optimization of 2D physics tables, for accuracy vs. memory size trade-off.
  - Multiple scattering models consolidation
    - Fix rare large unphysical scattering angles observed for small steps (ATLAS reported this), for Urban93 MSC model. Problem was due to parameterization applied out of validity range.
  - Built-in EM biasing
    - Introduction of brem-splitting option as built-in capability (ie, without the need for changing the physics list).
  - Optical photon
    - Dichroic Mirror
    - Optical properties in parallel worlds

Highlights of other new features in Geant4 version 10 (2)

- Hadronic physics
  - Introduction of isomers:
    - metastable states, with lifetime ≥ 1ns (user-tunable)
    - together with consistent evolution of
      - radioactive decay : decay may end-up with excited nucleus
      - photon evaporation : to generate γ lines consistent with isomers excitation energies ; if a fragment has lifetime > 1 μs, (user-tunable), it is tracked.
  - Bertini upgrade : New two-body angular distributions for gamma-N, pi-N and N-N
  - Nucleus-nucleus interactions possible in
    - FTF : ion-ion interaction introduced this year
      - From 3~4 GeV/A up to 100 GeV/A
      - Interfaced with Binary Cascade (for lower E) and PreCompound (even lower E)
    - INCL++
      - Retuned at low E, improved cross-sections for small clusters, validity extended to 10-15 GeV

Highlights of other new features in Geant4 version 10 (4)

- Hadronic physics (continued)
  - Muon capture
    - Improvements in decay-in-orbit, and cascades
  - Decommission of old GHEISHA-based LHEP models
  - Decomposition of CHIPS into granular modules and integration with other physics models
- Physics list
  - Decommission of obsolete or not-recommended physics lists.
  - Introducing “Physics Constructor”

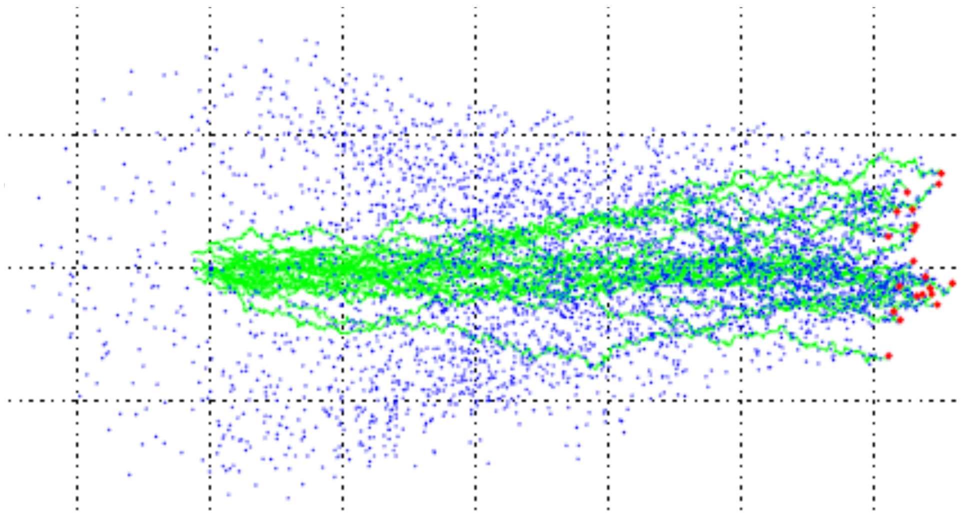
Highlights of other new features in Geant4 version 10 (4)

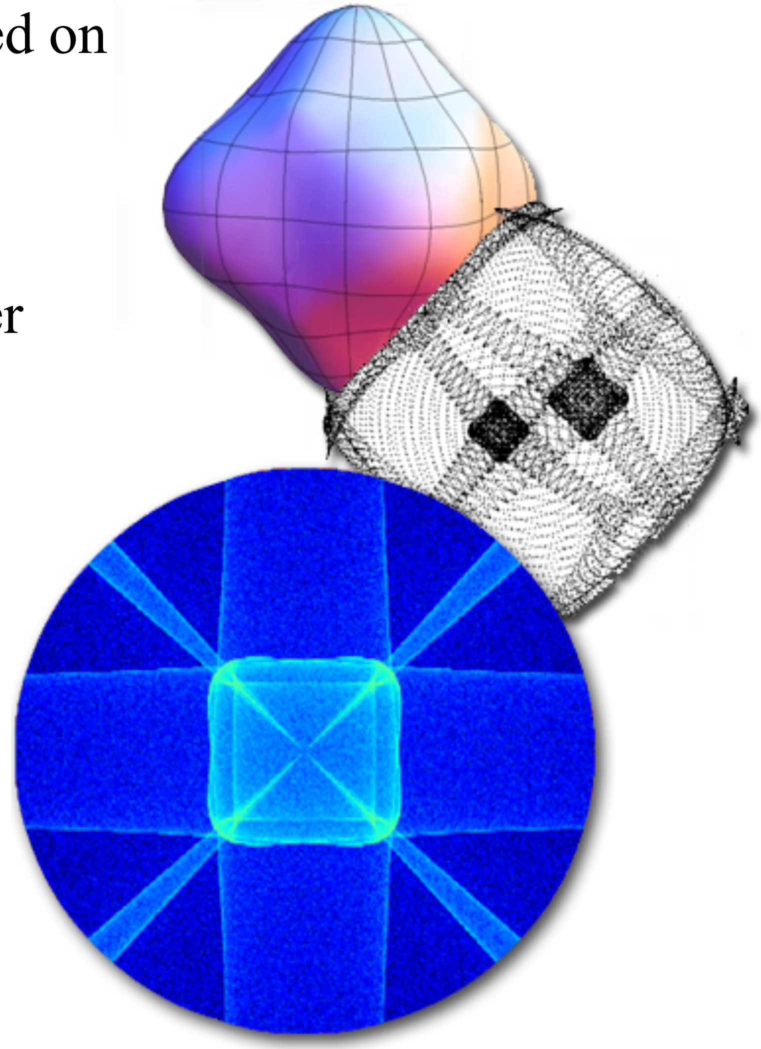
- Enhancements in biasing
  - Forced interaction, forced free flight
  - Improved handling of track weight for tracks that are not interacting
  - Improvements in usability
- Introduction of MT generates special issues for what concerns user's visualization:
  - User – application interaction has to be re-invented
    - Ie : if 200 threads, having 200 visualization windows makes no sense !
  - At 10.0 in multi-threaded mode, events could be drawn only after the event loop.
  - We anticipate innovative functionalities after release 10.0.
- Qt driver :
  - Certainly the most powerful GUI/vis driver in Geant4
    - Advantage of being portable
  - Qt5 !
  - Lot of functionalities developped, with tutorial at
    - http://geant4.in2p3.fr/spip.php?article84&lang=en
- New high-resolution transparent visualization
- We will drop GNUmake. Cmake will be the only supported installation build system.

Condensed Matter Physics in Geant4

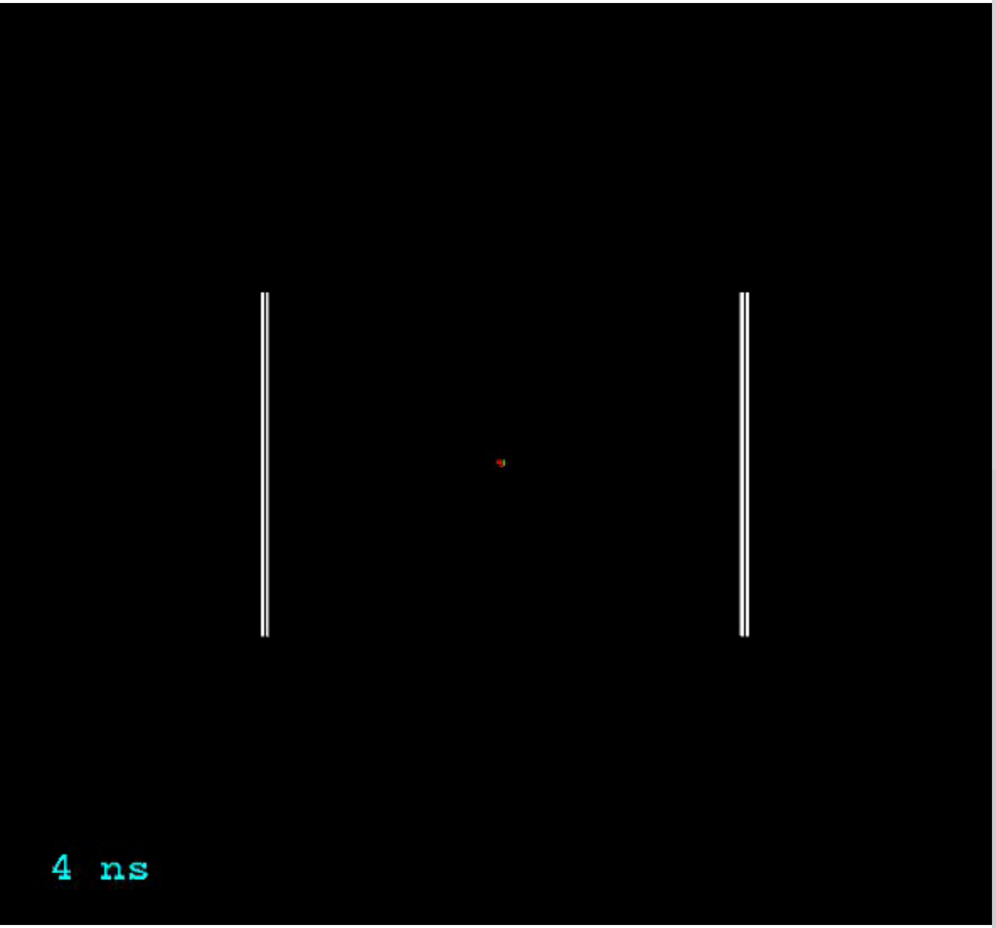
- Phonon propagation, including focusing based on elasticity tensor (right)

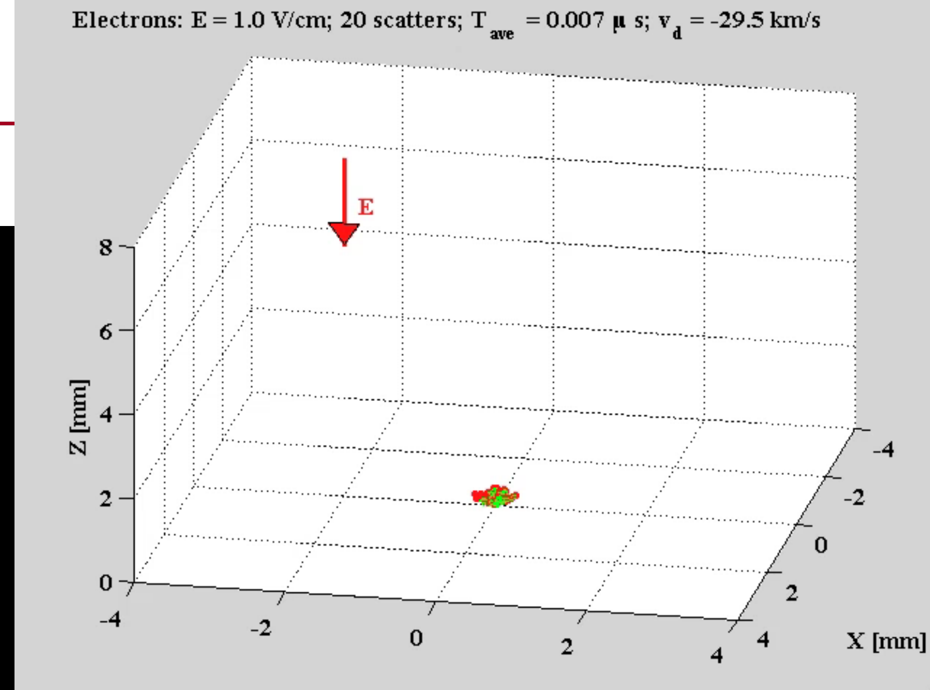
- e-/h+ transport, including conduction band anisotropy and Luke-Neganov emission, under development (below)

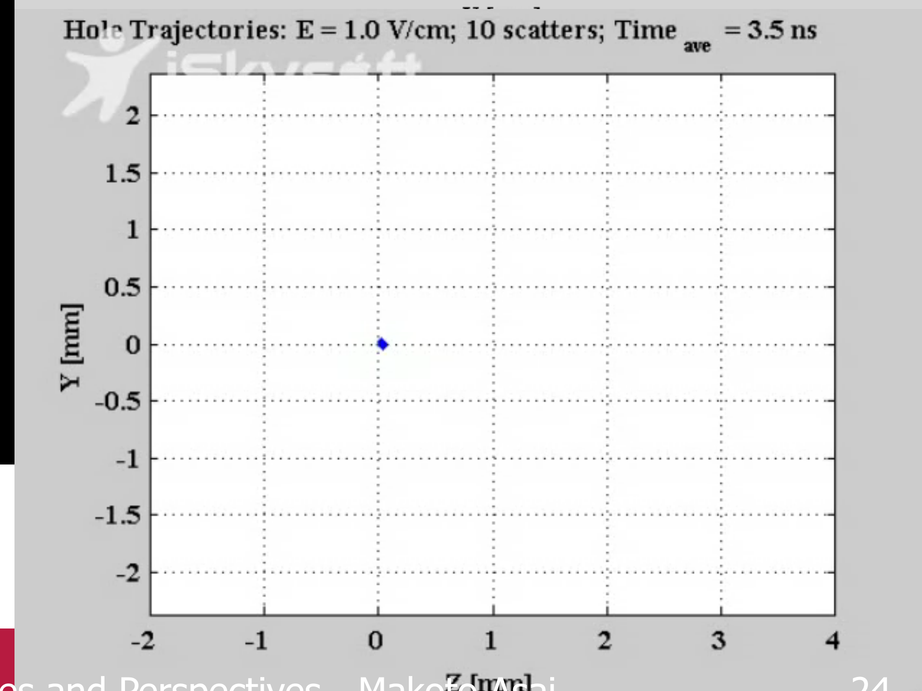




e-/h propagation with Luke phonon emission in Ge crystal

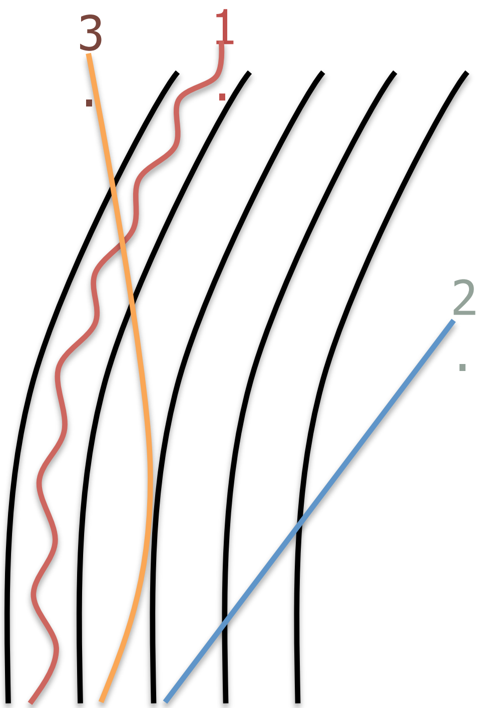


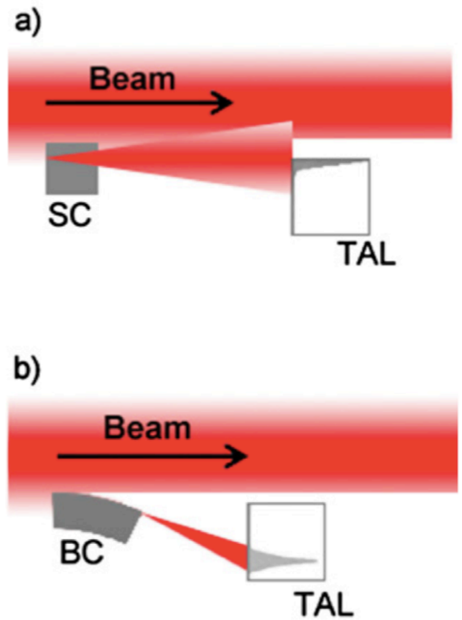


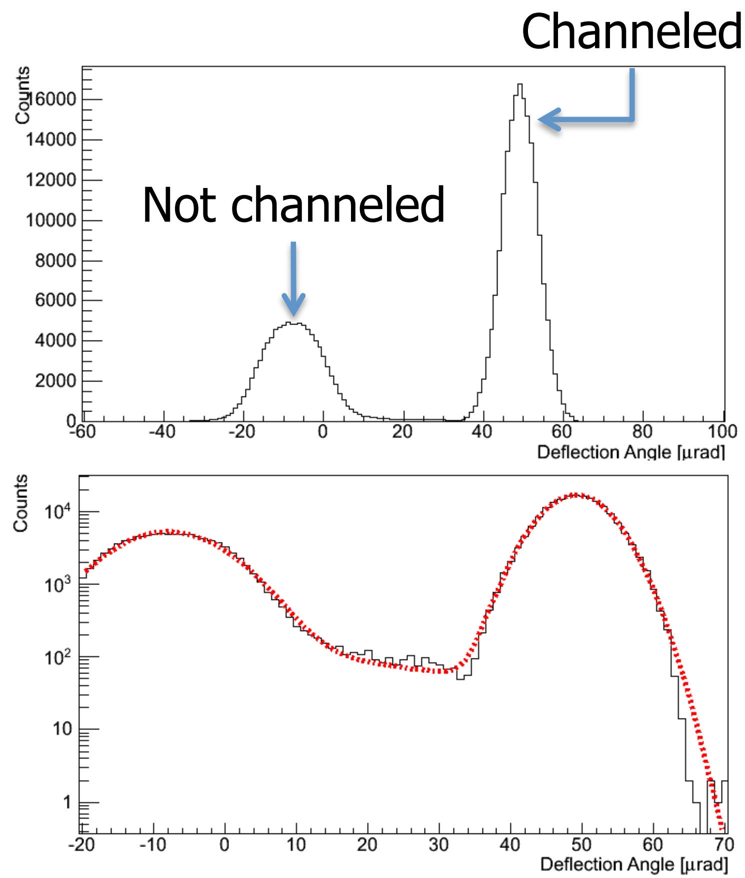


Bent crystal as a collimator

- Bent crystal can be used as a collimator to deflect particles of beam halo.
- This study will be extended for T-513 experiment at SLAC LCLS ESTB



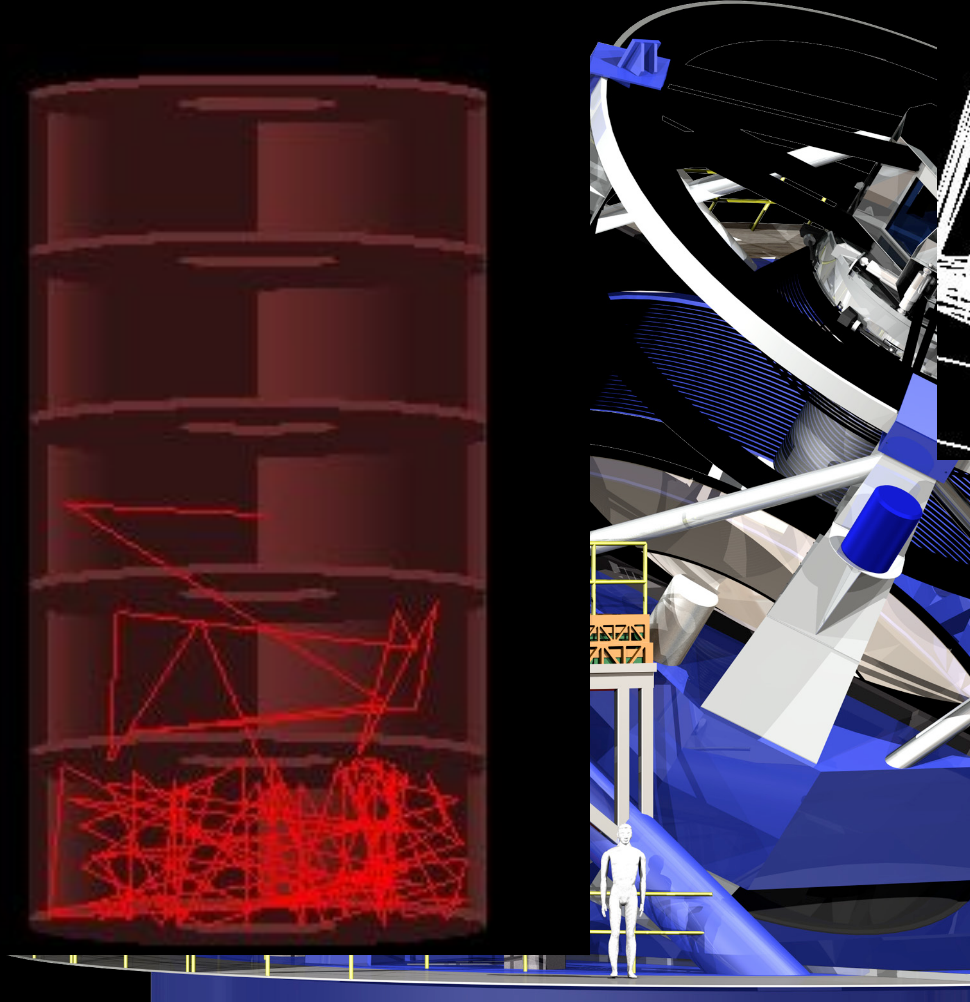


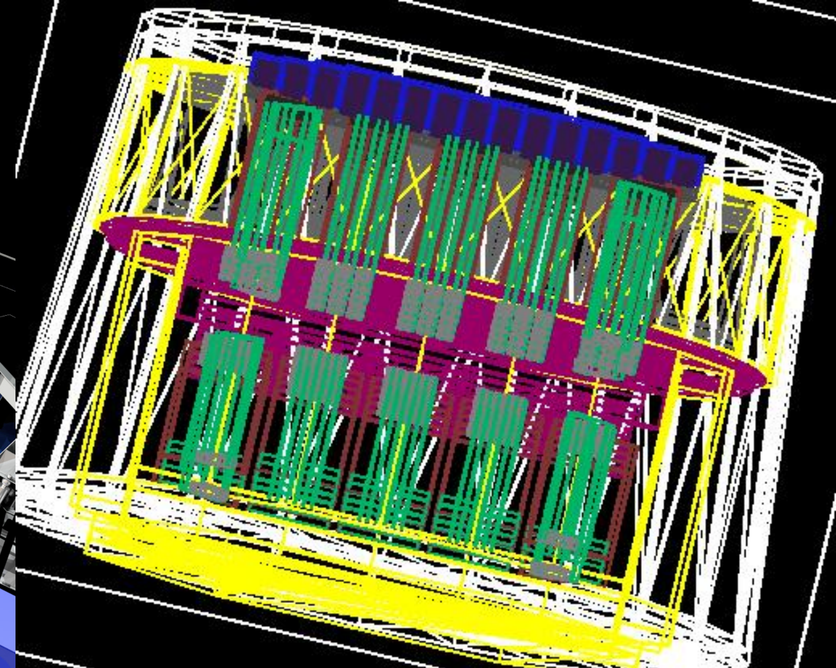


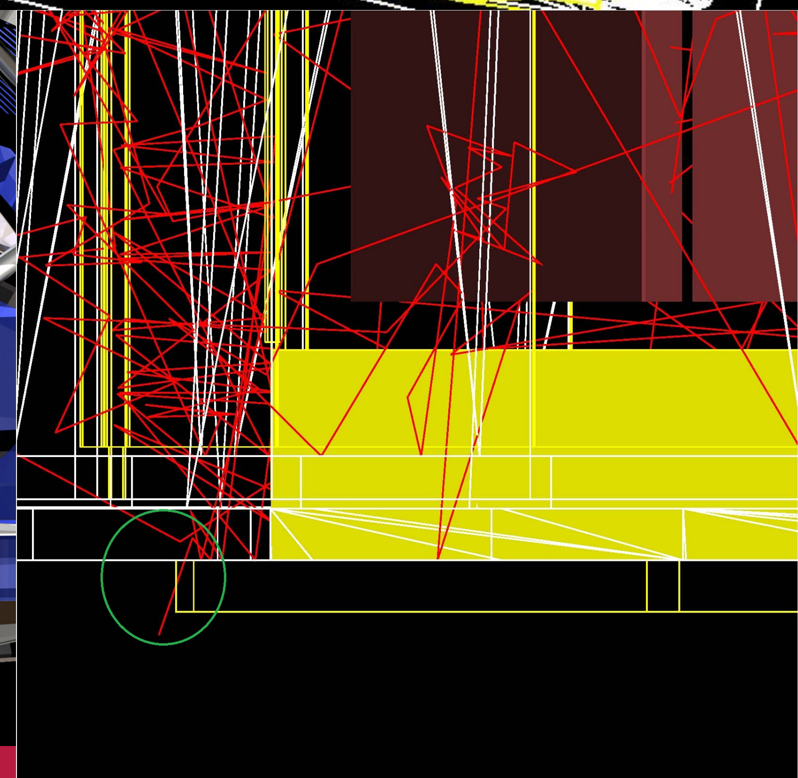
Enrico Bagli (INFN/Ferrara)

- W. Scandale et al., Phys. Lett. B 680 (2009) 129

LSST (Large Synoptic Survey Telescope)

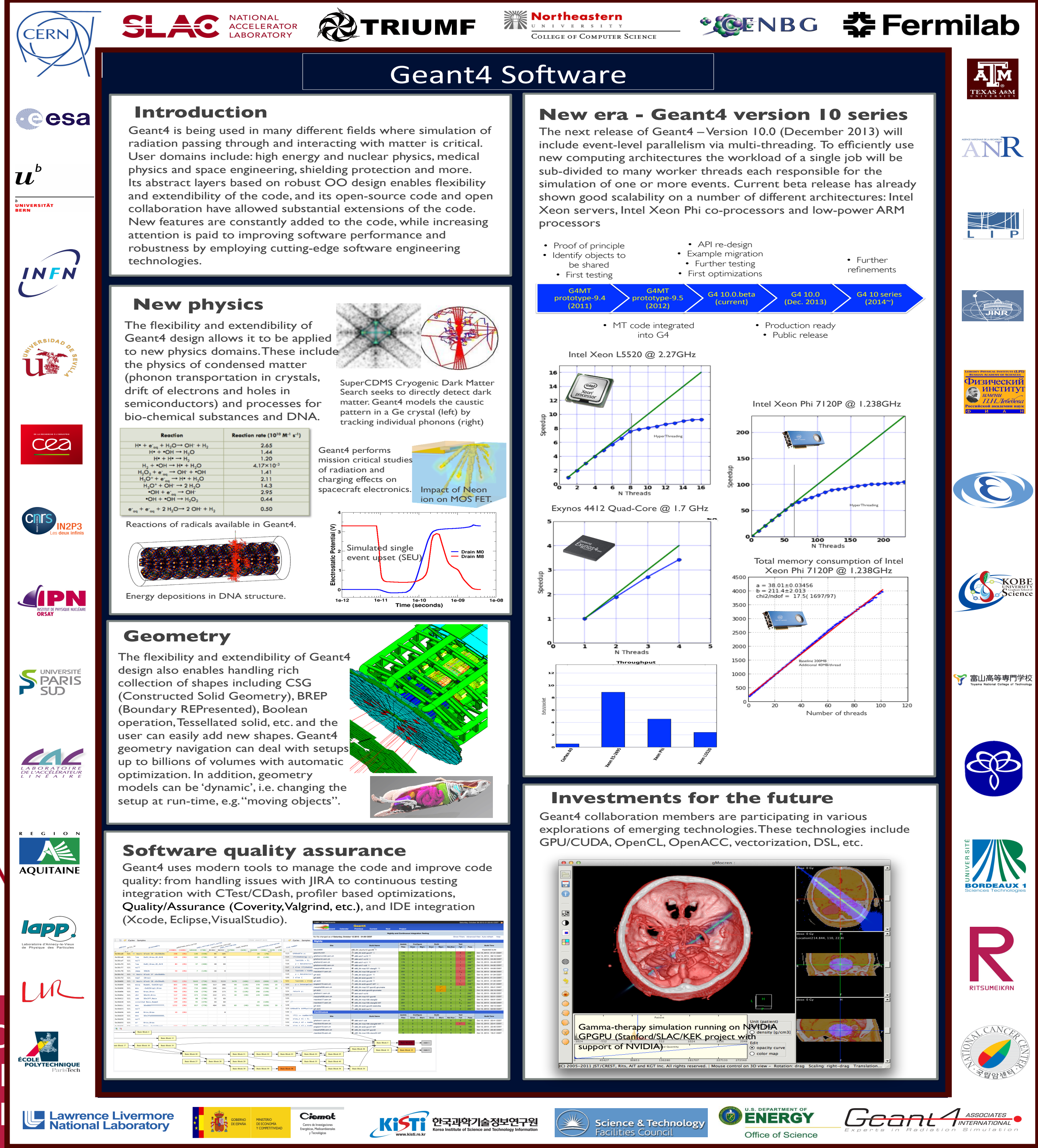






"Geant4 Applications for Modeling Molecular Transport in Complex Vacuum Geometries." J.Singal, J. Langton, R. Schindler, Int J Mod Sim Sci Comp, in press (arXiv:1302.2963)

Geant4
– the Future



Geant4 Software

Introduction

Geant4 is being used in many different fields where simulation of radiation passing through and interacting with matter is critical. User domains include: high energy and nuclear physics, medical physics and space engineering, shielding protection and more. Its abstract layers based on robust OO design enables flexibility and extendibility of the code, and its open-source code and open collaboration have allowed substantial extensions of the code. New features are constantly added to the code, while increasing attention is paid to improving software performance and robustness by employing cutting-edge software engineering technologies.

New physics

The flexibility and extendibility of Geant4 design allows it to be applied to new physics domains. These include the physics of condensed matter (phonon transportation in crystals, drift of electrons and holes in semiconductors) and processes for bio-chemical substances and DNA.

SuperCDMS Cryogenic Dark Matter Search seeks to directly detect dark matter. Geant4 models the caustic pattern in a Ge crystal (left) by tracking individual phonons (right)

Reaction	Reaction rate (10^{10} M^{-1} s^{-1})
$H\bullet + e^-_{aq} + H_2O \rightarrow OH^- + H_2$	2.65
$H\bullet + \bullet OH \rightarrow H_2O$	1.44
$H\bullet + H\bullet \rightarrow H_2$	1.20
$H_2 + \bullet OH \rightarrow H\bullet + H_2O$	4.17×10^{-3}
$H_2O_2 + e^-_{aq} \rightarrow OH^- + \bullet OH$	1.41
$H_3O^+ + e^-_{aq} \rightarrow H\bullet + H_2O$	2.11
$H_3O^+ + OH^- \rightarrow 2\,H_2O$	14.3
$\bullet OH + e^-_{aq} \rightarrow OH^-$	2.95
$\bullet OH + \bullet OH \rightarrow H_2O_2$	0.44
$e^-_{aq} + e^-_{aq} + 2\,H_2O \rightarrow 2\,OH^- + H_2$	0.50

Reactions of radicals available in Geant4.

Geant4 performs mission critical studies of radiation and charging effects on spacecraft electronics.

Impact of Neon ion on MOS FET.

Simulated single event upset (SEU)

Energy depositions in DNA structure.

Geometry

The flexibility and extendibility of Geant4 design also enables handling rich collection of shapes including CSG (Constructed Solid Geometry), BREP (Boundary REPresented), Boolean operation, Tessellated solid, etc. and the user can easily add new shapes. Geant4 geometry navigation can deal with setups up to billions of volumes with automatic optimization. In addition, geometry models can be 'dynamic', i.e. changing the setup at run-time, e.g. "moving objects".

Software quality assurance

Geant4 uses modern tools to manage the code and improve code quality: from handling issues with JIRA to continuous testing integration with CTest/CDash, profiler based optimizations, **Quality/Assurance (Coverity, Valgrind, etc.)**, and IDE integration (Xcode, Eclipse, VisualStudio).

New era - Geant4 version 10 series

The next release of Geant4 – Version 10.0 (December 2013) will include event-level parallelism via multi-threading. To efficiently use new computing architectures the workload of a single job will be sub-divided to many worker threads each responsible for the simulation of one or more events. Current beta release has already shown good scalability on a number of different architectures: Intel Xeon servers, Intel Xeon Phi co-processors and low-power ARM processors

- Proof of principle
- Identify objects to be shared
- First testing

G4MT prototype-9.4 (2011)

- API re-design
- Example migration
- Further testing
- First optimizations

G4MT prototype-9.5 (2012)

- MT code integrated into G4

G4 10.0.beta (current)

- Production ready
- Public release

G4 10.0 (Dec. 2013)

- Further refinements

G4 10 series (2014~)

Intel Xeon L5520 @ 2.27GHz

Intel Xeon Phi 7120P @ 1.238GHz

Exynos 4412 Quad-Core @ 1.7 GHz

Total memory consumption of Intel Xeon Phi 7120P @ 1.238GHz

Investments for the future

Geant4 collaboration members are participating in various explorations of emerging technologies. These technologies include GPU/CUDA, OpenCL, OpenACC, vectorization, DSL, etc.

Gamma-therapy simulation running on NVIDIA GPGPU (Stanford/SLAC/KEK project with support of NVIDIA)

Geant4 version 10 series

- The release in 2013 was a major release.
  - Geant4 version 10 – release date : Dec. 6, 2013
- The highlight is its multi-threading capability.
  - The world first large-scale physics software fully multithreaded
- Geant4 version 10 series will be evolving.
  - Performance improvements (both in physics and computing)
  - Missing functionalities yet to be migrated to multithreading,
  - Additional APIs
  - Additional functionalities
  - New physics

G4MT prototype-9.4 (2011) → G4MT prototype-9.5 (2012) → G4 10.0.beta (Jun.2013) → G4 10.0 (Dec. 2013) → G4 10 series (2014~)

G4MT prototype-9.4 (2011)
- Proof of principle
- Identify objects to be shared
- First testing

G4MT prototype-9.5 (2012)
- MT code integrated into G4

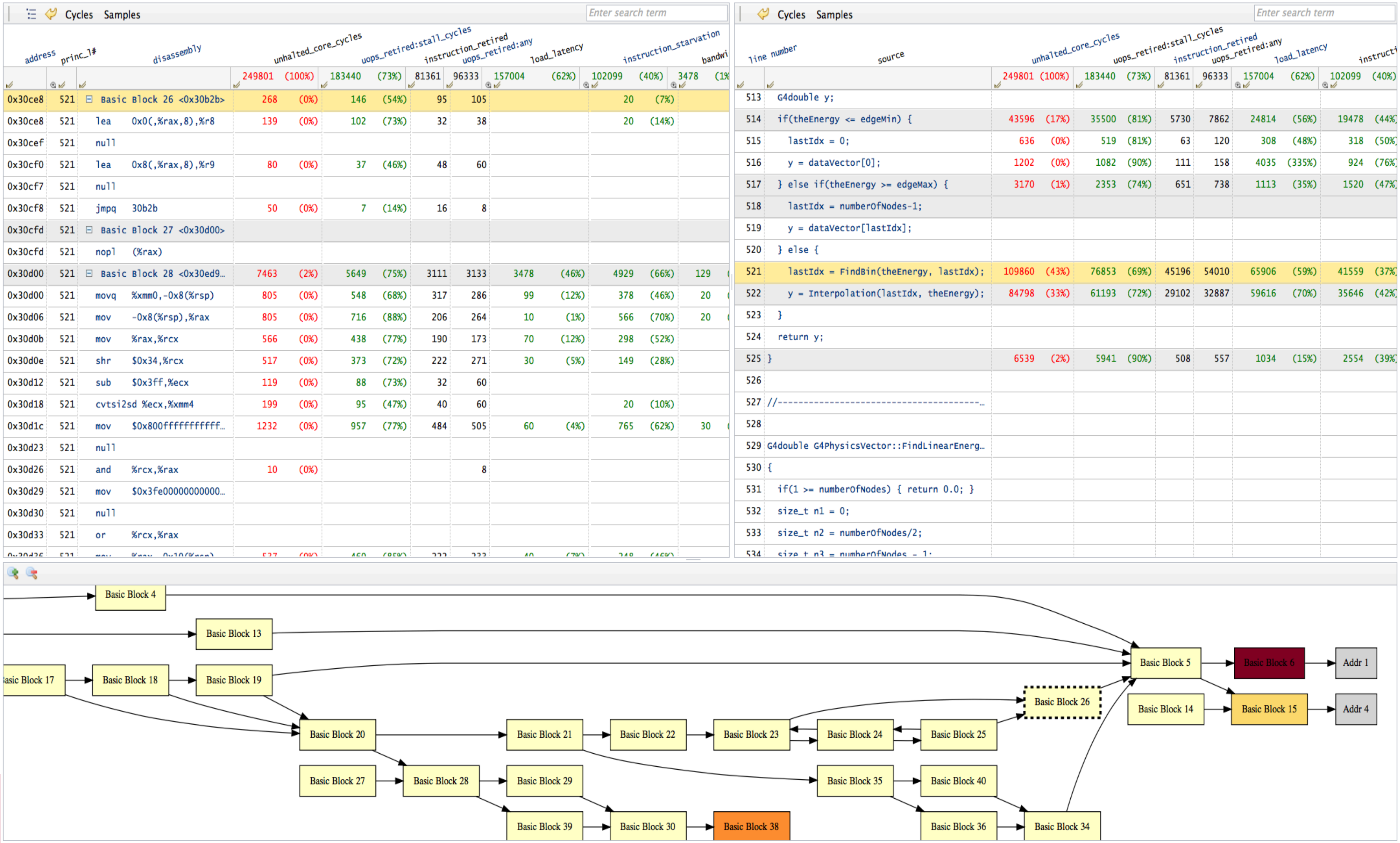
G4 10.0.beta (Jun.2013)
- API re-design
- Example migration
- Further testing
- First optimizations

G4 10.0 (Dec. 2013)
- Production ready
- Public release

G4 10 series (2014~)
- Further refinements

Software quality assurance (http://code.google.com/p/gooda/)

- SLAC is working with Google on performance measurements of Geant4-based application using Gooda tool, a PMU-based event data analysis package.

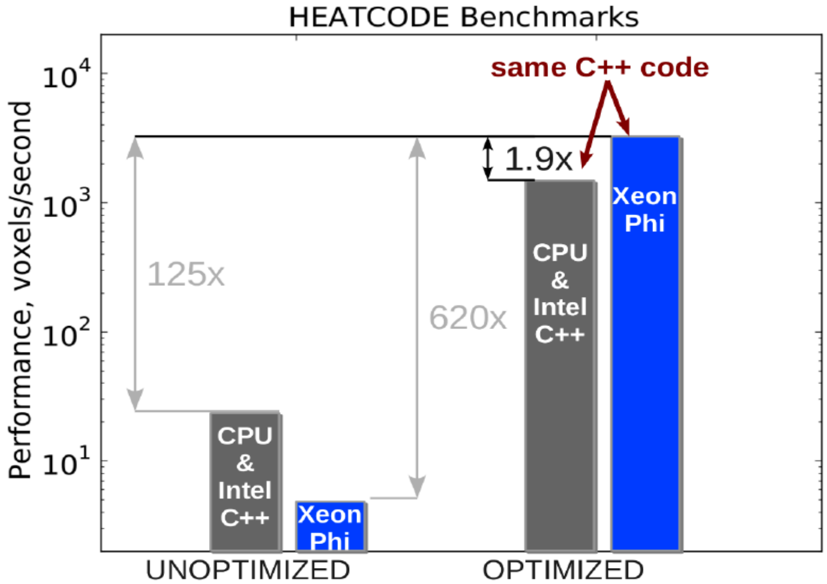


address	princ_l#	disassembly	unhalted_core_cycles		uops_retired:stall_cycles		instruction_retired	uops_retired:any	load_latency		instruction_starvation		bandwi
			249801	(100%)	183440	(73%)	81361	96333	157004	(62%)	102099	(40%)	3478 (1%
0x30ce8	521	Basic Block 26 <0x30b2b>	268	(0%)	146	(54%)	95	105			20	(7%)	
0x30ce8	521	lea 0x0(,%rax,8),%r8	139	(0%)	102	(73%)	32	38			20	(14%)	
0x30cef	521	null											
0x30cf0	521	lea 0x8(,%rax,8),%r9	80	(0%)	37	(46%)	48	60					
0x30cf7	521	null											
0x30cf8	521	jmpq 30b2b	50	(0%)	7	(14%)	16	8					
0x30cfd	521	Basic Block 27 <0x30d00>											
0x30cfd	521	nopl (%rax)											
0x30d00	521	Basic Block 28 <0x30ed9…	7463	(2%)	5649	(75%)	3111	3133	3478	(46%)	4929	(66%)	129
0x30d00	521	movq %xmm0,-0x8(%rsp)	805	(0%)	548	(68%)	317	286	99	(12%)	378	(46%)	20
0x30d06	521	mov -0x8(%rsp),%rax	805	(0%)	716	(88%)	206	264	10	(1%)	566	(70%)	20
0x30d0b	521	mov %rax,%rcx	566	(0%)	438	(77%)	190	173	70	(12%)	298	(52%)	
0x30d0e	521	shr $0x34,%rcx	517	(0%)	373	(72%)	222	271	30	(5%)	149	(28%)	
0x30d12	521	sub $0x3ff,%ecx	119	(0%)	88	(73%)	32	60					
0x30d18	521	cvtsi2sd %ecx,%xmm4	199	(0%)	95	(47%)	40	60			20	(10%)	
0x30d1c	521	mov $0x800ffffffffff…	1232	(0%)	957	(77%)	484	505	60	(4%)	765	(62%)	30
0x30d23	521	null											
0x30d26	521	and %rcx,%rax	10	(0%)				8					
0x30d29	521	mov $0x3fe00000000000…											
0x30d30	521	null											
0x30d33	521	or %rcx,%rax											

line number	source	unhalted_core_cycles		uops_retired:stall_cycles		instruction_retired	uops_retired:any	load_latency		instructi	
		249801	(100%)	183440	(73%)	81361	96333	157004	(62%)	102099	(40%)
513	G4double y;										
514	if(theEnergy <= edgeMin) {	43596	(17%)	35500	(81%)	5730	7862	24814	(56%)	19478	(44%
515	lastIdx = 0;	636	(0%)	519	(81%)	63	120	308	(48%)	318	(50%
516	y = dataVector[0];	1202	(0%)	1082	(90%)	111	158	4035	(335%)	924	(76%
517	} else if(theEnergy >= edgeMax) {	3170	(1%)	2353	(74%)	651	738	1113	(35%)	1520	(47%
518	lastIdx = numberOfNodes-1;										
519	y = dataVector[lastIdx];										
520	} else {										
521	lastIdx = FindBin(theEnergy, lastIdx);	109860	(43%)	76853	(69%)	45196	54010	65906	(59%)	41559	(37%
522	y = Interpolation(lastIdx, theEnergy);	84798	(33%)	61193	(72%)	29102	32887	59616	(70%)	35646	(42%
523	}										
524	return y;										
525	}	6539	(2%)	5941	(90%)	508	557	1034	(15%)	2554	(39%
526											
527	//-------------------------------------…										
528											
529	G4double G4PhysicsVector::FindLinearEnerg…										
530	{										
531	if(1 >= numberOfNodes) { return 0.0; }										
532	size_t n1 = 0;										
533	size_t n2 = numberOfNodes/2;										

Performance Expectations: "Two Birds with One Stone"

- Performance will be disappointing if code is not optimized for multi-core CPUs
- Optimized code runs better on the MIC platform *and* on the multi-core CPU
- Single code for two platforms + Ease of porting = Incremental optimization



More information in case study on research.colfaxinternational.com

Dr. Andrey Vladimirov (Colfax International) – seminar at Stanford U.

Optimization Example: In-Place Square Matrix Transposition

```
#pragma omp parallel for
  for (int i = 0; i < n; i++) { // Distribute across threads
    for (int j = 0; j < i; j++) { // Employ vector load/stores
      const double c = A[i*n + j]; // Swap elements
      A[i*n + j] = A[j*n + i];
      A[j*n + i] = c;
    }
  }
```

Unoptimized code:

- Large-stride memory accesses
- Inefficient cache use
- Does not reach memory bandwidth limit

Dr. Andrey Vladimirov (Colfax International) – seminar at Stanford U.

Code optimization for both CPU and coprocessor

Tiling a Parallel For-Loop (Matrix Transposition)

```
#pragma omp parallel for
  for (int ii = 0; ii < n; ii += TILE) { // Distribute across threads
    const int iMax = (n < ii+TILE ? n : ii+TILE); // Adapt to matrix shape
    for (int jj = 0; jj <= ii; jj += TILE)  { // Tile the work
      for (int i = ii; i < iMax; i++) { // Universal microkernel
        const int jMax = (i < jj+TILE ? i : jj+TILE); // for whole matrix
#pragma loop count avg(TILE) // Vectorization tuning
#pragma simd // Vectorization hint
        for (int j = jj; j<jMax; j++) { // Variable loop count (bad)
          const double c = A[i*n + j]; // Swap elements
          A[i*n + j] = A[j*n + i];
          A[j*n + i] = c;
    } } } }
```

Better (but not optimal) solution:

- Loop tiling to improve locality of data access
- Not enough outer loop iterations to keep 240 threads busy

Dr. Andrey Vladimirov (Colfax International) – seminar at Stanford U.

Further Optimization: Code Snippet

```
#pragma omp parallel
  {
#pragma omp for schedule(guided)
    for (int k = 0; k < nTilesParallel; k++) { // Bulk of calculations here
      const int ii = plan[HEADER_OFFSET + 2*k + 0]*TILE; // Planned order
      const int jj = plan[HEADER_OFFSET + 2*k + 1]*TILE; // of operations
      for (int j = jj; j < jj+TILE; j++) { // Simplified main microkernel
#pragma simd // Vectorization hint
#pragma vector nontemporal // Cache traffic hint
        for (int i = ii; i < ii+TILE; i++) { // Constant loop count (good)
          const double c = A[i*n + j]; // Swap elements
          A[i*n + j] = A[j*n + i];
          A[j*n + i] = c;
        } } }
    // Transposing the tiles along the main diagonal and edges...
    // ...
```

- Longer code but still in the C language; works for CPU and MIC

Dr. Andrey Vladimirov (Colfax International) – seminar at Stanford U.

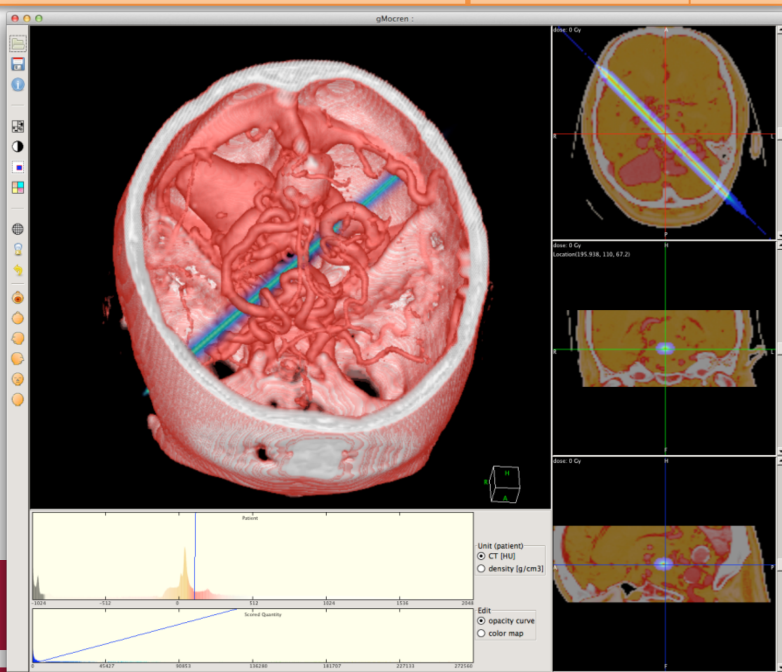
New physics – new opportunities

- Neutrino interactions
  - Should come with enriched event biasing options
- Electron/hole drift in semiconductor
- More phonon physics
- Channeling effects
- Physics with crystal structure in general
  - X-ray diffraction
- Single atom irradiation
- Target material polarization
- Chemical reactions of radicals in DNA-scale
- New domains ?

- Note : Geant4 kernel is robust enough over 20 years of evolution. This stability enables risk-free extensions to new physics.

Investment for the future
– Low energy EM physics ported to GPU

Primary	Phantom	Time/History CPU (sec)	Time/History GPU (sec)	CPU/GPU
20 MeV electron	Water	1.06E-03	2.52E-05	42.1
20 MeV electron (e-spread)	Lung	1.20E-03	2.67E-05	44.9
20 MeV electron (e-spread)	Bone	9.76E-4	2.54E-05	38.4
6 MeV photon	Water	4.47E-04	1.12E-05	39.9
6 MV photon (e-spread)	Lung	3.52E-04	9.16E-06	38.4
6 MV photon (e-spread)	Bone	3.59E-04	9.00E-06	39.9
18 MV photon (e-spread)	Lung	4.05E-04	1.12E-05	36.2
18 MV photon (e-spread)	Bone	4.29E-04	1.17E-05	36.7



Observed GPU speed up over a single-thread CPU: ~40x

Left: Irradiation of 50 million 6 MeV monochromatic photons calculated by GPGPU (not for real treatment use, demonstration purposes only !)

Collaboration of SLAC, Stanford ICME and KEK with support of NVIDIA

To sum up

- Geant4 is a general purpose Monte Carlo simulation tool for elementary particles passing through and interacting with matter. It finds quite a wide variety of user domains including high energy and nuclear physics, space engineering, medical applications, material science, radiation protection and security.
- This year is the 20th year anniversary of Geant4. After 20 years with several architectural evolutions, Geant4 is still steadily evolving.
  - Latest evolution was Geant4 version 10.0 released in December 2013 that is the first fully multithreaded large-scale physics software in the world.
- Given Geant4 is nowadays mission-critical for many users including all LHC experiments, space missions, medical applications, etc., Geant4 is to be kept maintained and still evolving for at least next decade.