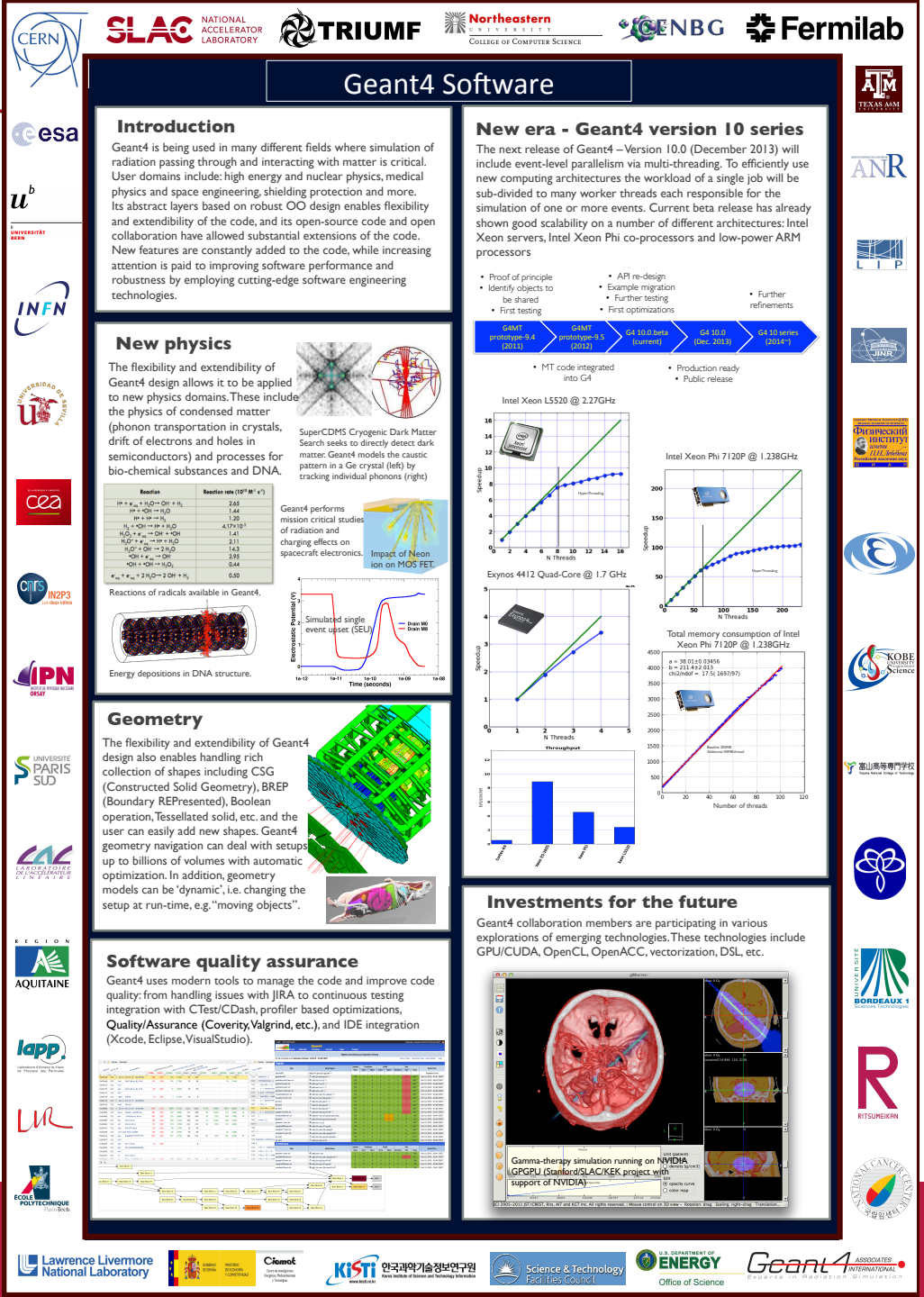
2014 Development Plan
- non-physics part -

Makoto Asai (SLAC)
On behalf of the Geant4 Collaboration
Geant4 Technical Forum
March 20, 2014

SLAC NATIONAL ACCELERATOR LABORATORY
U.S. DEPARTMENT OF ENERGY
Office of Science

Contents

- Full set of proposed 2014 work plan can be found at http://geant4.web.cern.ch/geant4/support/planned_features.shtml
- In this talk I will summarize some key development items in non-physics part of the work plan.
- Your suggestions, comments, requirements are essential.



Geant4 version 10 series

- The release in 2013 was a major release.
  - Geant4 version 10.0 – 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, (s)he may install Geant4 in sequential mode.

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

- MT code integrated into G4

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

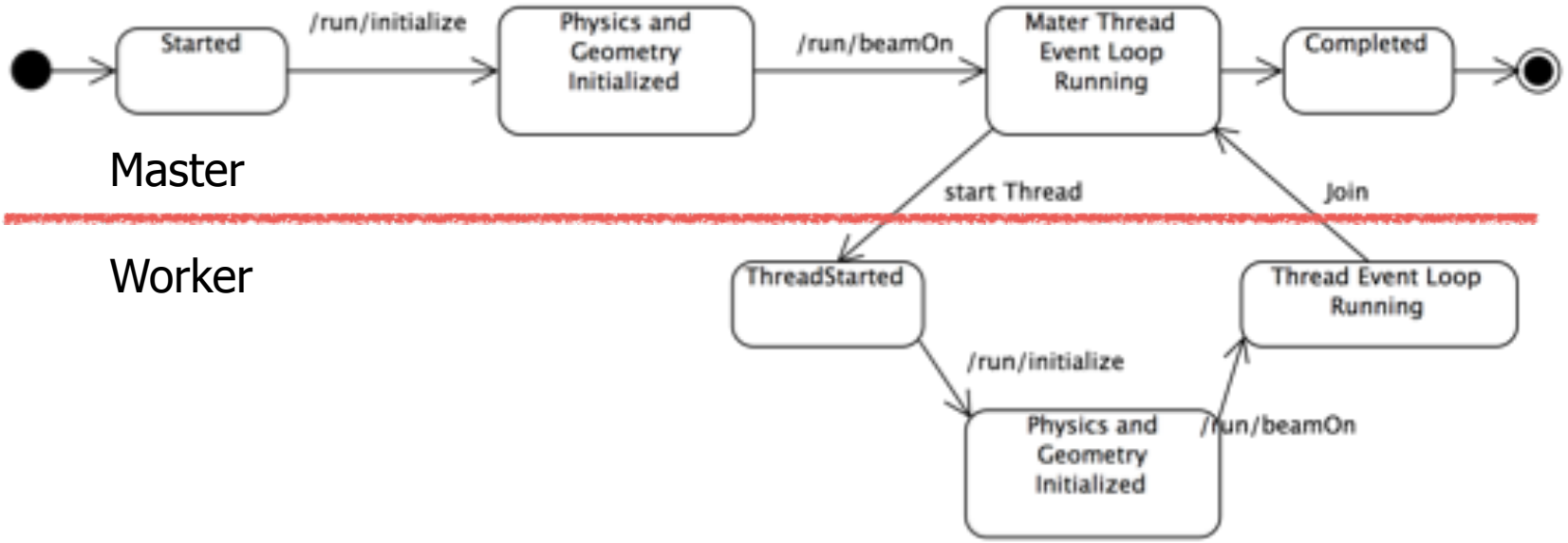
- API refinements
- Production ready
- Public release

- Further refinements and optimizations

We appreciate your feedbacks !

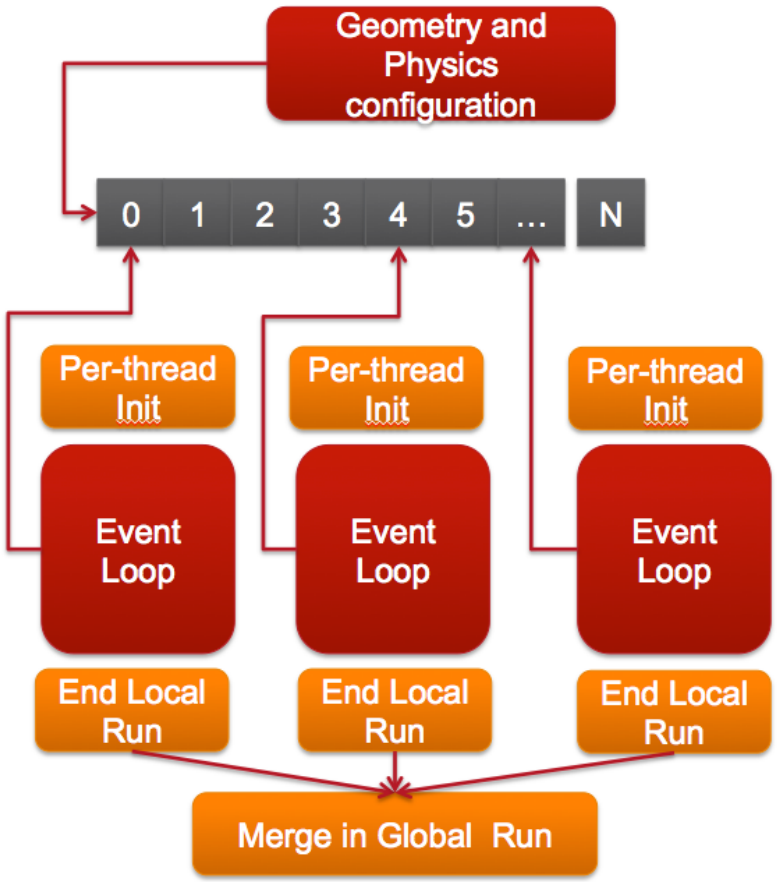
Simplified Master / Worker Model

- A G4 (with MT) application can be seen as simple finite state machine
- Threads do not exists before first /run/beamOn
- When master starts the first run spawns threads and distribute work



Event tasking is not round robin

- During the master event loop, an event (or a bunch of events) is tasked to a worker thread in first-come-first-served basis.
  - To minimize the latency at the end of master event loop
  - Required toward our next goal of complete decoupling between the master event loop and worker thread initialization/termination
    - Desirable for TBB-based simulation (see later slides)
- Master thread generates all the necessary initial seeds for all events and dispatch.
  - For the sake of full reproducibility regardless of number of threads.



Per-event seeds pre-prepared in a "queue"

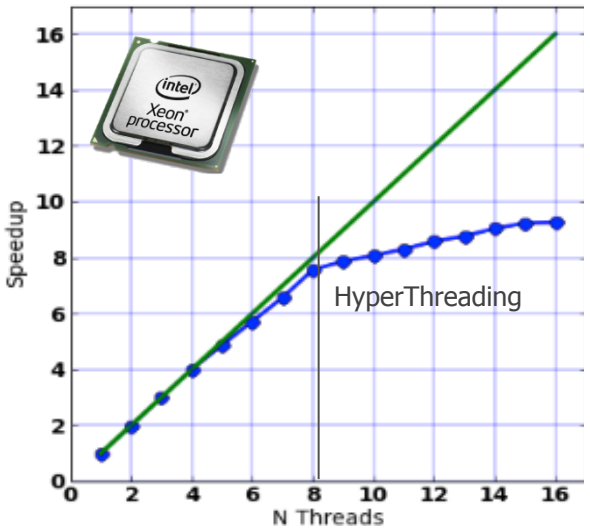
Threads compete for next event to be processes

Command line scoring and G4tools automatically merge results from threads

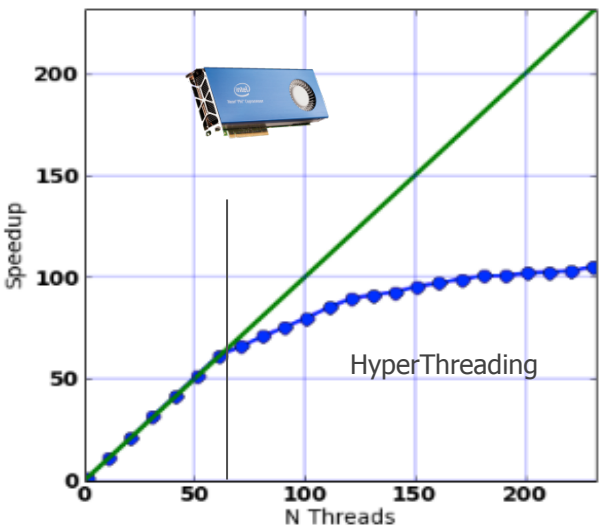
Performance on different architectures

- Current 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.

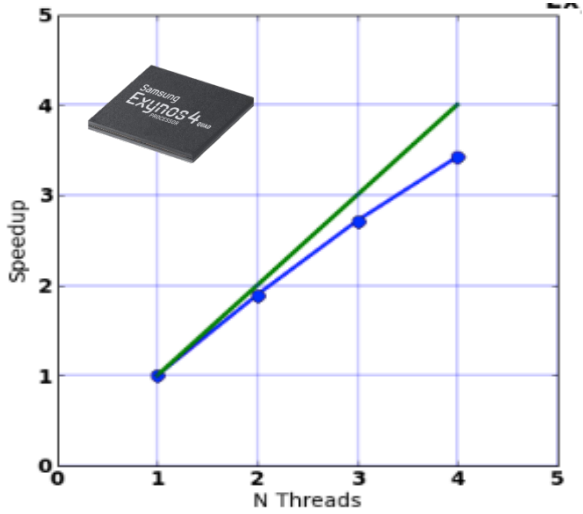
Intel Xeon L5520 @ 2.27GHz



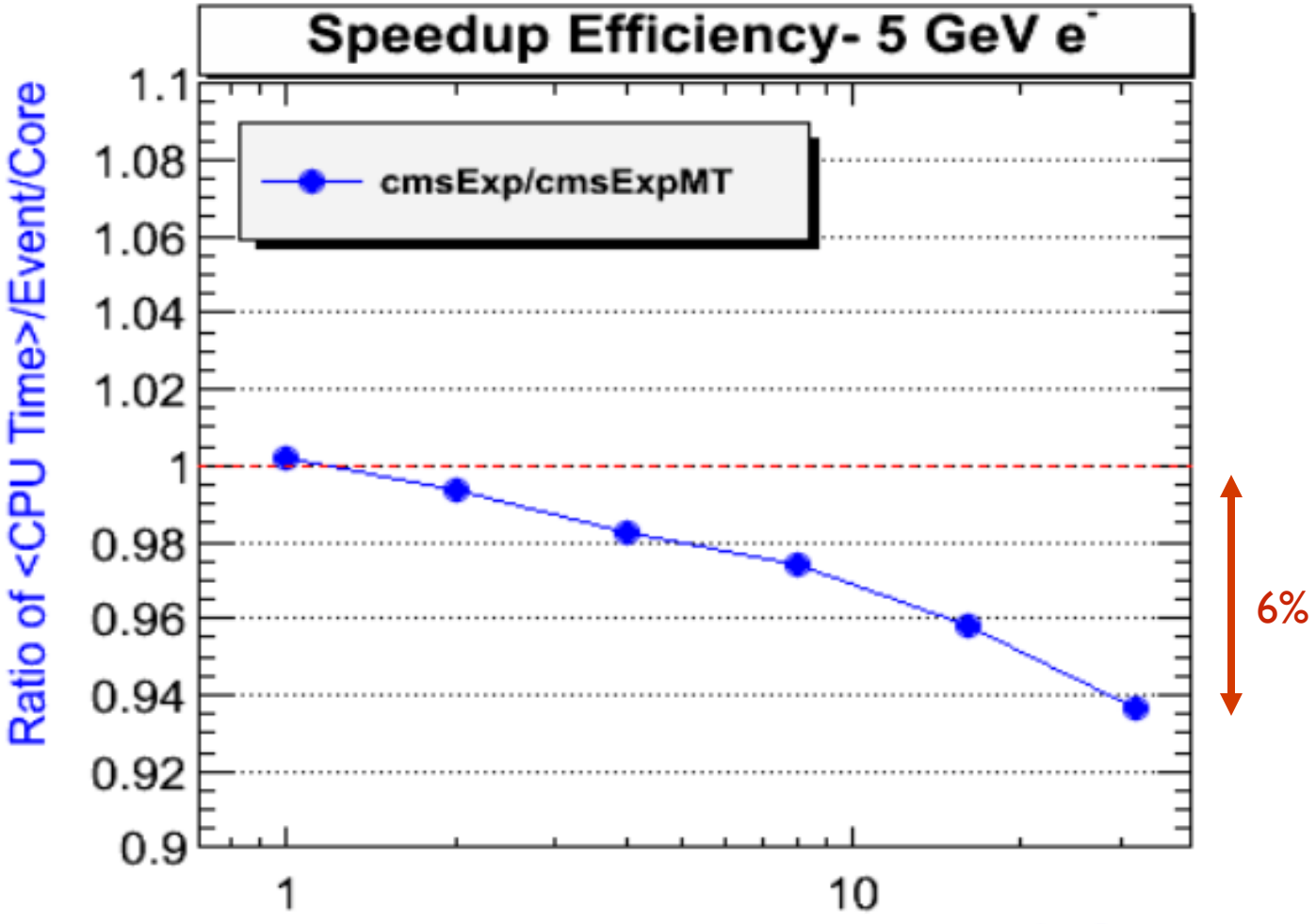
Intel Xeon Phi 7120P @ 1.238GHz



Exynos 4412 Quad-Core @ 1.7 GHz



Comparison with sequential mode (speed)

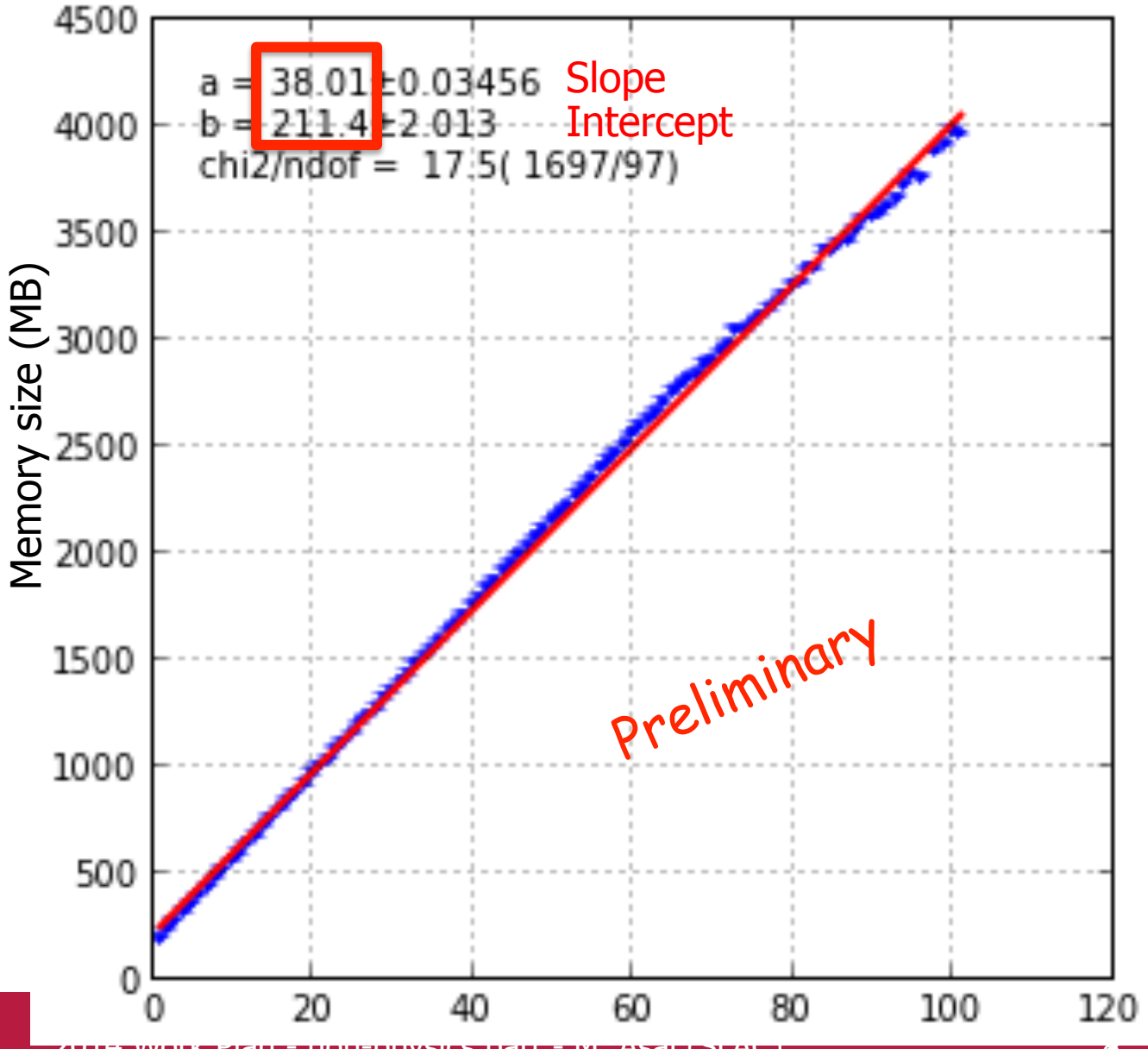


"CMS-ish" geometry

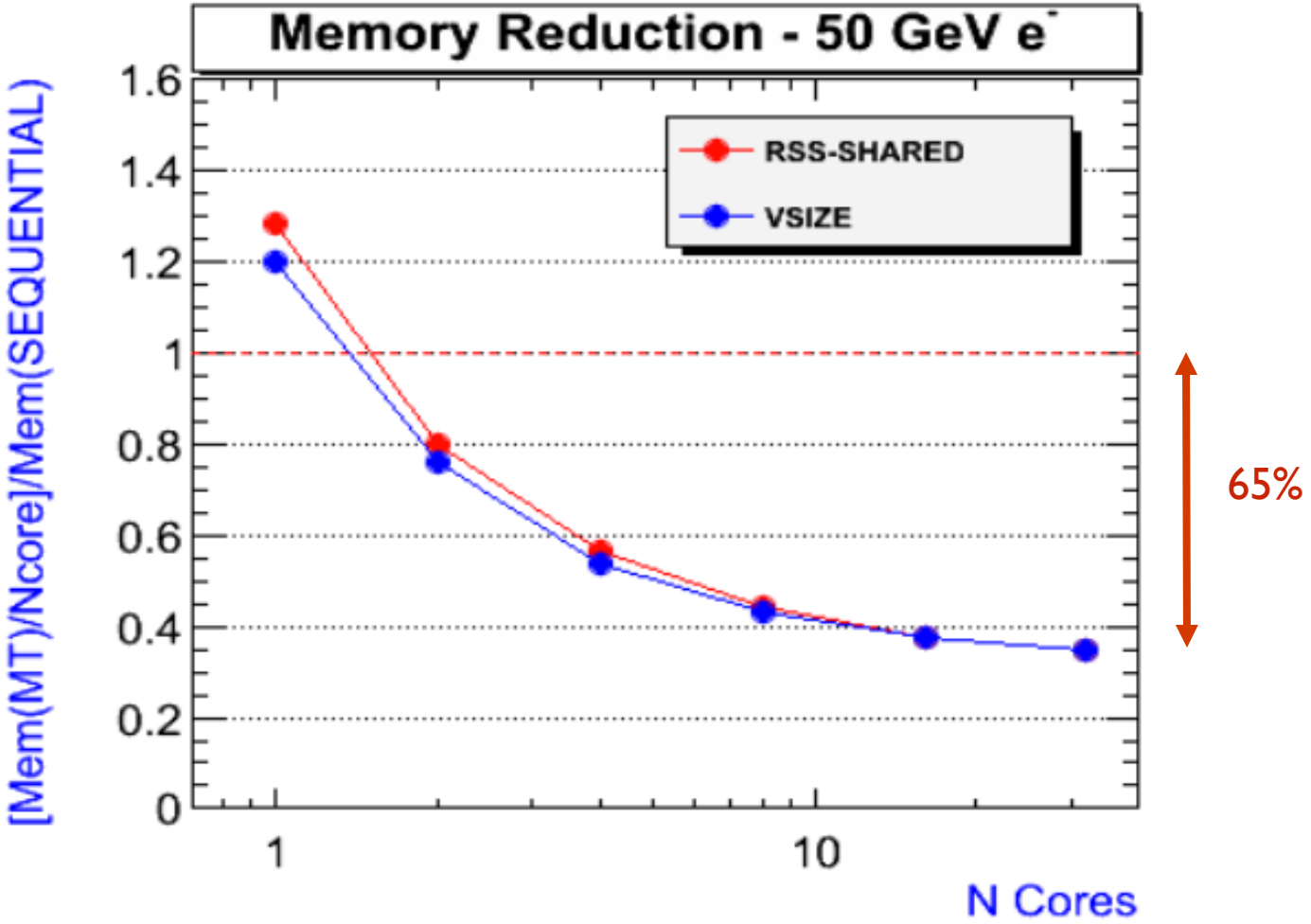
Processor: AMD Opteron(tm) Processor 6128 : 32 cores (4 CPU sockets x 8 cores)
CPU: 2000 MHz, Cache: 512 KB, Total Memory: 66007532 kB
OS: Linux kernel 2.6.32-358.11.1.el6.x86_64 GCC 4.4.6 20120305 (Red Hat 4.4.6-4)

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 thread-local arrays in physics processes



Comparison with sequential mode (memory)

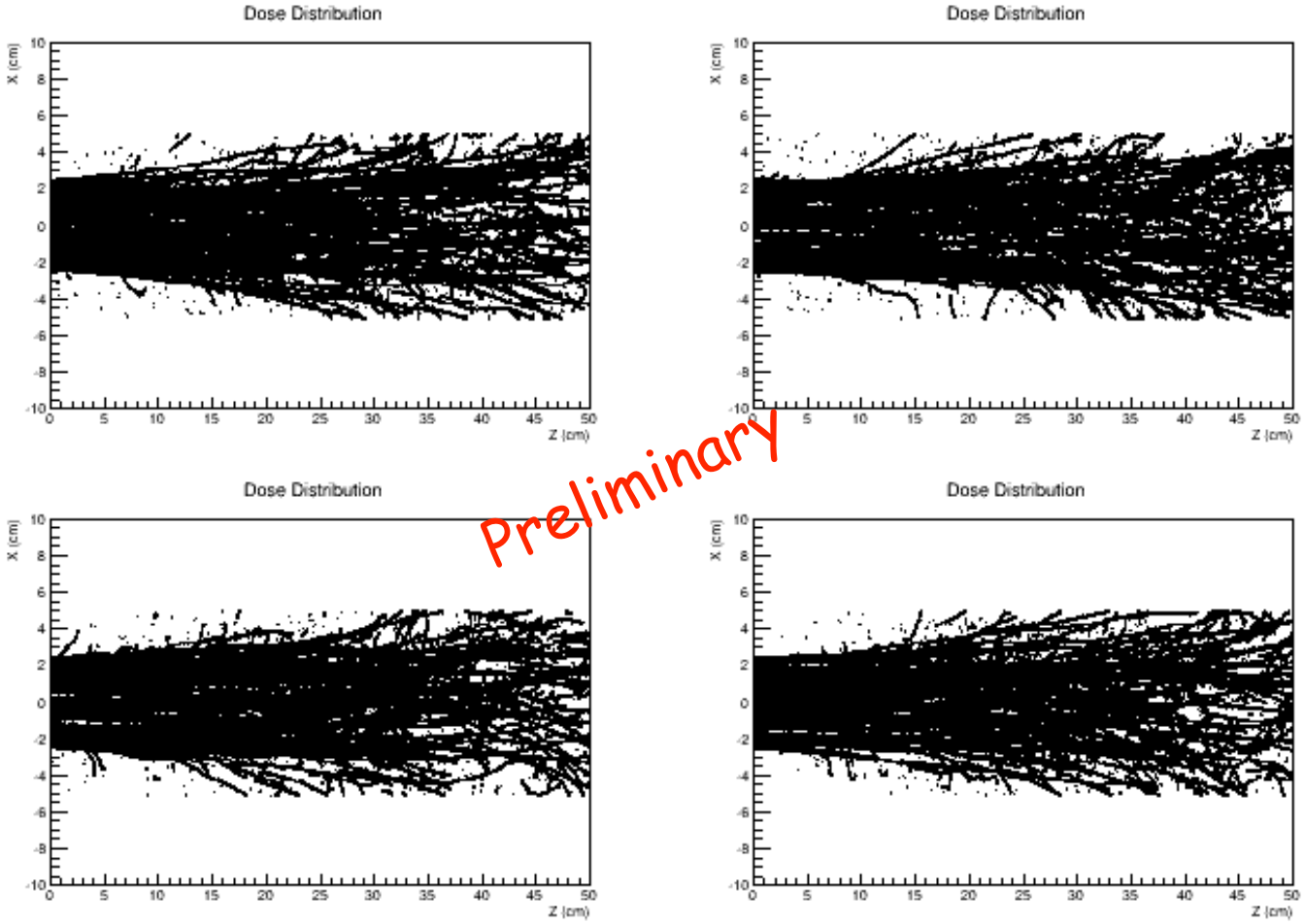


"CMS-ish" geometry

Processor: AMD Opteron(tm) Processor 6128 : 32 cores (4 CPU sockets x 8 cores)
CPU: 2000 MHz, Cache: 512 KB, Total Memory: 66007532 kB
OS: Linux kernel 2.6.32-358.11.1.el6.x86_64 GCC 4.4.6 20120305 (Red Hat 4.4.6-4)

MPI + multi-threading

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



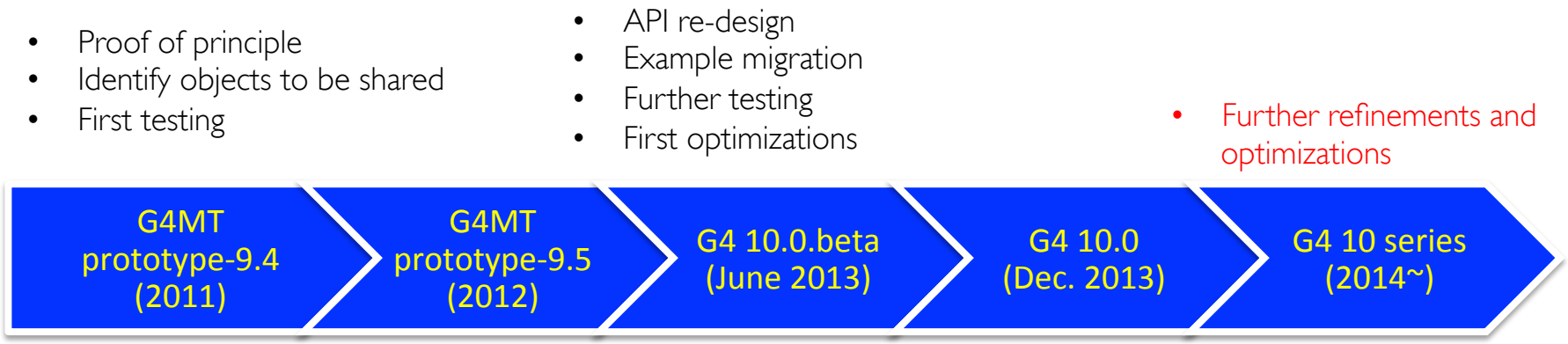
- 4 MPI processes with 2 cores each
- Each MPI process owns histogram
- Threads merge dose calculation in shared histogram

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 G4 v10 can be used in a TBB-based application where TBB-tasks are responsible for simulating events.
  - We didn't need to modify any concrete G4 class/method to adapt to TBB.
- We provide an example in version 10.0 release to demonstrate the way of integrating Geant4 with TBB.
- We keep investigating where/how to reduce memory use.
- We will keep communicating with our users to polish our top-level interfaces.
  - Next step includes decoupling of master event loop and worker thread initialization/termination.

Beyond Geant4 version 10.0

- Geant4 version 10 series will be evolving.



- Performance improvements
  - Algorithm optimization / local vectorization
    - without losing code readability / maintainability / flexibility
  - Optimization of file access
- Memory space reduction in particular for per-thread memory
  - Sharing more physics vectors and other objects among threads
- Multithreading leftover
  - Some visualization, neutron_hp, general particle source, Geant4e, etc.
- Completion of decoupling between master event loop and worker thread initialization / termination
  - See later slide

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

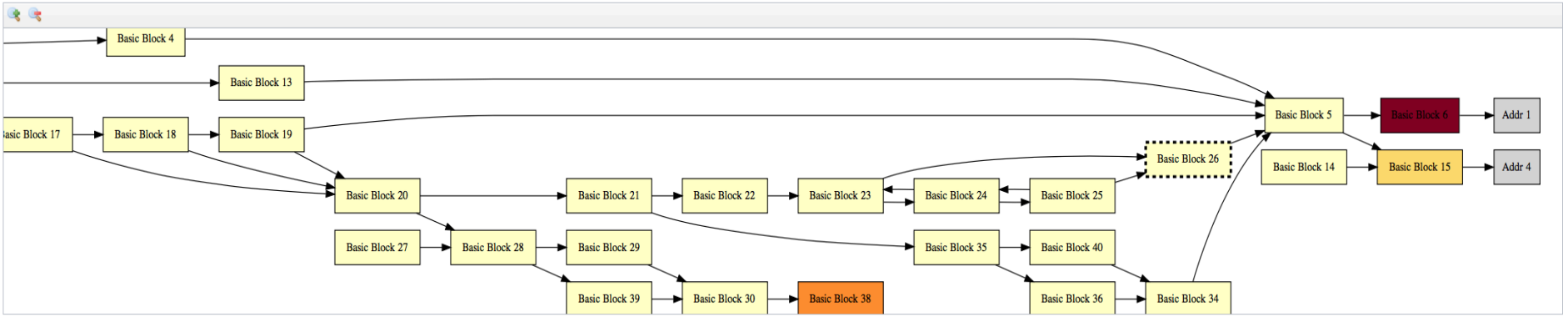
- We are working with Google on performance measurements of Geant4-based application using Gooda tool, a PMU-based event data analysis package.

Cycles Samples — Enter search term

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 $0x800ffffffffffff…	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												

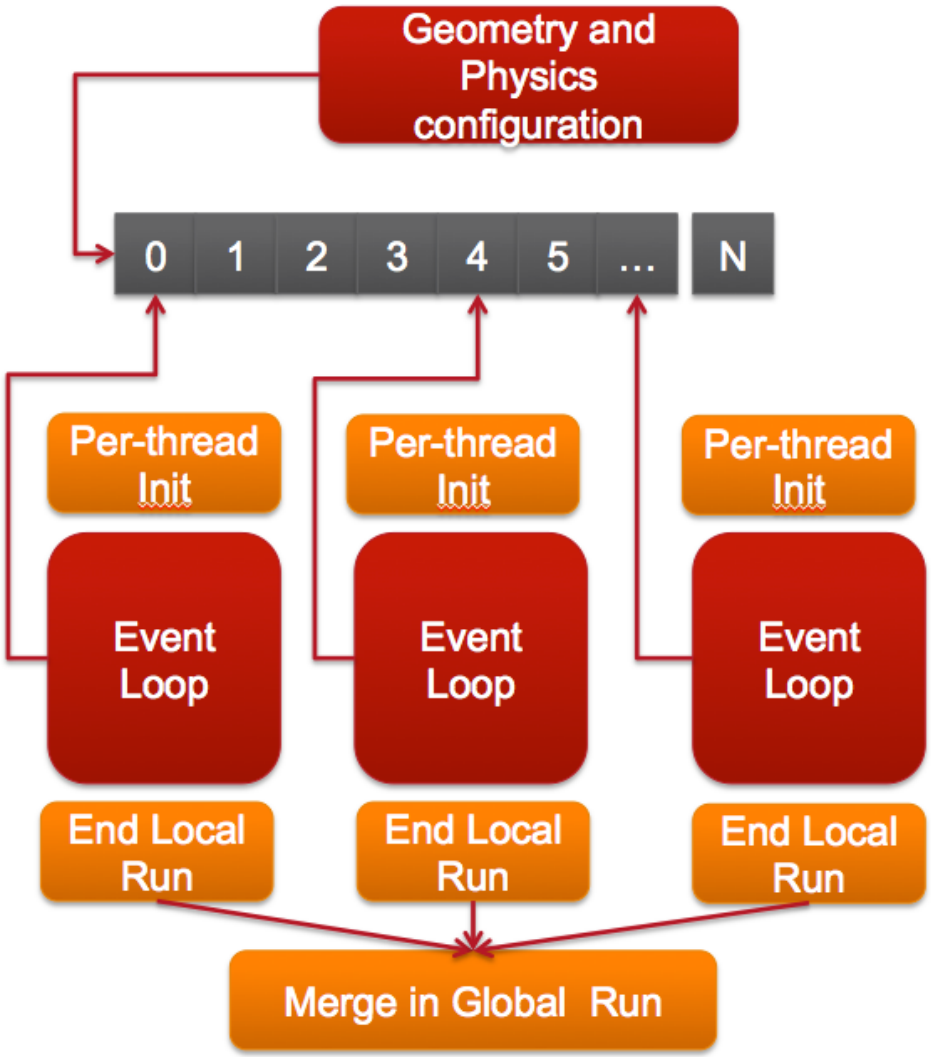
Cycles Samples — Enter search term

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;										
534	size_t n3 = numberOfNodes - 1;										



Decoupling of master event loop and worker thread initialization/termination

- Ensuring a worker thread to join at any time during the event loop of the master
  - After the master thread finishes initialization for geometry and cross-section tables to be shared
- Ensuring a worker thread to leave at any time during the event loop of the master
  - After finishing assigned task (an event or a bunch of events)
- We plan to complete this decoupling with some additional APIs
  - First set of new APIs will come with v10.1-beta in June.
  - Your feedbacks are essential.



Geometry, Persistency

- Geometry
  - Identification of first/last step in a volume for curved tracks
  - Consolidate use of precise ComputeSafety() in navigation
  - Complete implementation of the unified solids library
  - Refactor navigator to separate thread-dependent state
  - Enable parameterization by solids type in MT mode

- Persistency
  - Extension of GDML reader/writer to support multi-union constructs
  - Add support for parallel geometries to ASCII format
  - Support for Ntuple columns of vector
  - Addition of read function to analysis module
  - Support for 1D, 2D profiles in analysis manager
  - Batch plotting

Generic Processes (Event Biasing)

- Generic processes
  - Generic Biasing
    - Enrich event biasing options: Splitting; leading particle biasing; alternative importance biasing
    - Biasing of charged particles; occurrence biasing; DXTRAN-like biasing; material/isotope biasing; Woodcock tracking
  - Geometry Biasing & Importance
    - Automatic smart importance biasing assignment
  - Reverse Monte Carlo
    - Improvements of EM processes for case of thick shielding
    - Complete migration to multi-threading

Interfaces, Visualization

- Interfaces
  - Improvements to Qt driver interface
  - Web application interface based on Wt
  - Support of multi-threading for G4Py

- Visualization
  - Addition of a Visualization Management Thread to see events in MT mode before run is complete
  - Support of user-drawn primitives in multi-threaded mode
  - New driver OGLFile to produce image files in batch jobs where there is no graphics card present
  - Consolidation of Wt driver for web browser visualization
  - Support save and restore viewpoint and save and replay fly-through in OpenInventor
  - Development of visualization solutions for iOS and Android devices
  - New Transparent Visualization tool to support high resolution transparent visualization with ability to rotate and zoom
  - Integrated visualization of field lines (electric, magnetic, ...)

Examples, Documentation

- Basic & Extended Examples
  - Complete migration to Geant4 native analysis tools
  - New hadronic example demonstrating Fission Fragment model
  - Complete migration to multithreading for feasible examples

- Advanced examples
  - Migration of selected examples to multi-threading
  - Migration to Geant4 native analysis tools
  - New advanced example about medical liver therapy with Y90 micro-spheres

- Documentation
  - Complement missing/insufficient contexts
    - Thread control for advanced users
    - QMD, neutron_hp and some other physics models
  - Expand Web pages of publications, validation results, and external citations