Business from technology

Serpent 2 – Status and future plans

Jaakko Leppänen
VTT Technical Research Centre of Finland

Outline

- Background
- Memory issues in Serpent – the main reason for re-writing the code
- New features in new version:
  - Optimization modes and new approach to parallelization
  - Photon physics
  - Variance reduction
- Current status of the project and future plans:
  - What has been done so far
  - Distribution schedule and beta-testing
  - Future plans

Background

- Development of Serpent 2 started in September 2010, under working title “Super-Serpent”
- Reasons for re-writing the code:
  - Development of Serpent 1 has been carried out for over six years without any “grand vision” on how things should be done as a whole
  - This live-and-learn approach has lead to overly-complicated calculation routines and hundreds or even thousands of lines of redundant source code
  - Adding new features, while keeping everything together, becomes increasingly complicated
  - *Excessive memory usage brings serious limitations to burnup calculation and parallelization*

Background

- After some consideration, it was decided that the problems in Serpent 1 are best solved by starting everything from scratch:
  - Simplified and better structured coding without anything extra
  - Opportunity to do things the way they should have been done in the first place
  - Implementation of new features (gamma transport, etc.) can be taken into account from the beginning
  - <u>*More emphasis on memory management, parallelization and super-computing applications (hence the name)*</u>
- Some parts of source code (physics) can be taken from Serpent 1 without major modifications

Memory issues in Serpent

- Serpent 1 is optimized for performance in *lattice physics* applications at the cost of memory usage:
  - Microscopic reaction cross sections are reconstructed on a unionized energy grid → grid search needs to be performed only once, each time the neutron scatters to a new energy
  - Macroscopic cross sections are pre-calculated before transport cycle → no need to sum over material compositions
- And in burnup calculation mode:
  - One-group transmutation cross sections are calculated using the spectrum-collapse method → no need to tally reaction rates during transport cycle

Memory issues in Serpent

- Advantages:
  - Considerable savings in total CPU time
  - Calculation of macroscopic cross sections is easy due to the use of a single energy grid
  - Calculation of majorant cross section for delta-tracking is easy due to the use of a single energy grid
  - Unionized energy grid is a natural choice as the energy bin structure for the spectrum-collapse method (easy to implement, maximum resolution)

  But most of all: *Serpent running time is almost independent of the number of nuclides or materials in the problem* → *ideal for burnup calculation problems*

Memory issues in Serpent

- Drawbacks:
  - Reconstruction of cross sections requires a lot of memory for storing redundant data points
  - Grid thinning, if used, results in the loss of data
  - Memory demand per material increases to tens of megabytes → number of burnable materials is limited to a few hundred
  - Memory demand in MPI mode is multiplied by the number of parallel tasks → severe limitations in parallelization capability

Memory issues in Serpent

- Memory issues and limitations are almost exclusively related to burnup calculation
- The capabilities of Serpent 1 are more or less sufficient for 2D assembly burnup calculations, where the number of depletion zones is ~100.
- But what about:
  - 3D assembly burnup calculations – adding a new dimension easily multiplies the number of depletion zones?
  - Research reactors – thousands of depletion zones?
  - Power reactors – tens or hundreds of thousands of depletion zones?
- And what about development of computer capacity – tens or hundreds of CPU cores that cannot be used in calculation due to excessive memory usage?

Memory issues in Serpent

- Specific goals in the development of “Super”:
  - Capability to handle at least tens of thousands of depletion zones in burnup calculation (if required)
  - Capability to run smaller burnup calculation problems as efficiently as Serpent 1
  - Capability to perform parallel calculation without limitations (“Super-computing”)
- These goals are achieved by:
  - Different levels of optimization depending on problem size
  - Shared memory techniques for parallel calculation

Optimization modes

- The options to balance performance and memory usage are the same as in Serpent 1:
  - Reconstruction of microscopic cross sections on the unionized energy grid – affects total memory usage
  - Calculation of macroscopic total cross sections – affects memory usage per burnable material
  - Spectrum-collapse method for burnup calculation – affects memory usage per burnable material
  - Generation of pre-defined reaction lists to speed up summation over material-wise totals – affects memory usage per burnable material, but becomes significant only in very large problems

[TOTAL MEMORY USAGE] ≈ [STORAGE SPACE FOR MICROSCOPIC XS] + [NUMBER OF MATERIALS] × [STORAGE SPACE PER MATERIAL]

Optimization modes

- The use of these options is divided into five optimization modes:

Mode	Reconstructed microxs	Pre-calculated macroxs	Material-wise reaction lists	Spectrum-collapse in burnup mode	Group constant generation
0	-	-	-	-	-
1	-	-	YES	-	-
2	YES	-	YES	YES	-
3	-	YES	YES	YES	YES
4	YES	YES	YES	YES	YES

- Group constant calculation involves tallying macroscopic reaction rates, so the option is switched off in modes 0 – 2, in which the corresponding cross sections are not pre-calculated.

Optimization modes

- Each mode is designed for a slightly different purpose:

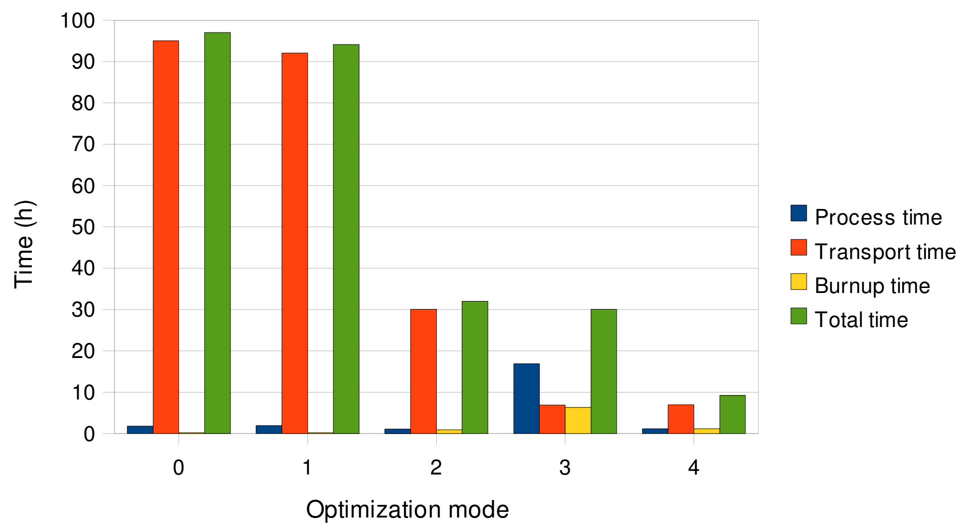
Mode	Description	To be used for
4	Maximum performance at the cost of memory usage	2D lattice physics applications similar to Serpent 1 – group constant generation and assembly burnup calculations involving less than 100 depletion zones
3	Fast transport cycle with lower memory demand	Similar to mode 4, but to be used when memory size is a limitation, not well suited for large burnup calculation problems due to long processing time per material
2	Good performance in larger burnup calculation problems	Burnup calculations involving hundreds of depletion zones, poor performance for group constant generation
1	Minimized memory demand at the cost of performance	Very large burnup calculation problems involving thousands of depletion zones
0	No optimization	Burnup calculation problems that are too large for mode 1, reference for other modes

- NOTE: these modes and options are still preliminary, and everything depends on the computing environment

Optimization modes - example

- Example case:
    - 17 x 17 PWR assembly burnup calculation with burnable absorber pins, irradiated to 40 MWd/kgU burnup
    - 66 burnable material regions
    - 42 depletion steps with predictor-corrector calculation
    - Concentrations of 1300 nuclides tracked (300 with cross sections), 1290 transmutation reactions
    - 3 million neutron histories per cycle (500 active cycles of 6000 neutrons)
- Calculation repeated in modes 0-4
- Single-CPU calculation, 3.47 GHz, Intel Xeon workstation, 46 G memory

Optimization modes - example



Mode	Process time (h)		Transport time (h)		Burnup time (h)		Total time (h)	
0	1.8	(1.6)	95.1	(13.7)	0.1	(0.1)	97.0	(10.5)
1	1.9	(1.7)	92.1	(13.3)	0.1	(0.1)	94.1	(10.2)
2	1.1	(1.0)	30.0	(4.3)	0.8	(0.7)	32.0	(3.5)
3	16.9	(15.4)	6.8	(1.0)	6.3	(5.4)	30.1	(3.3)
4	1.1	(1.0)	6.9	(1.0)	1.2	(1.0)	9.2	(1.0)
Serpent 1.1.16	N/A		6.7	(1.0)	N/A		9.1	(1.0)

Optimization modes - example

- No energy grid unionization for microscopic xs in mode 3 → calculation of material totals takes (processing) time
- 7-8 minutes spent in solving the Bateman equations, time not dependent on optimization mode
- Number of nuclides and transmutation reactions can probably be reduced without compromising accuracy → reduction in transport calculation time when spectrum collapse method is not used (modes 0 and 1)
- Processing and burnup calculation time could be reduced by optimizing the routines?
- Calculation of majorant cross section may become a problem when the number of materials increases to several thousand (use conservative estimates?)

Optimization modes - example

- Memory demand depends on optimization mode:
  - Mode 0: 146 M total, 0.2 M per material → potential for ~190,000 depletion zones
  - Mode 1: 170 M total, 0.6 M per material → potential for ~ 70,000 depletion zones
  - Mode 2: 5898 M total, 3.8 M per material → potential for ~ 8,000 depletion zones
  - Mode 3: 2423 M total, 33 M per material* → potential for ~1,000 depletion zones
  - Mode 4: 7014 M total, 20 M per material → potential for ~1,500 depletion zones
- Serpent 1.1.16 uses about 8808 M total / 43 M per material

* Grid thinning doesn't work well in mode 3 → larger grid size and memory demand per material compared to mode 4

Parallelization – MPI

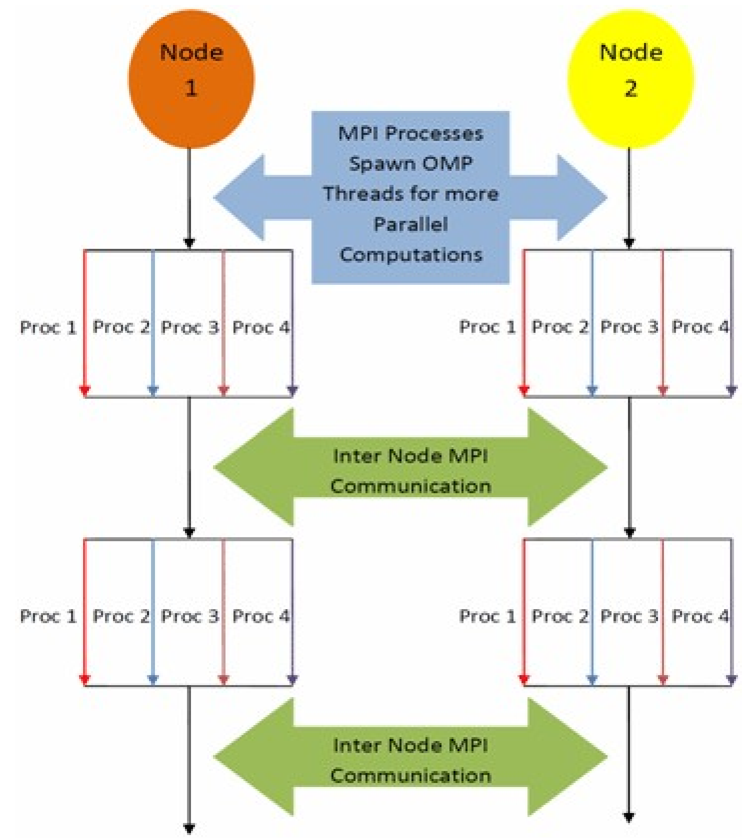
- Parallelization of the transport loop in Serpent 1 is based on the Message Passing Interface (MPI):
  - Each parallel task receives a copy of all input data
  - Population size is divided by the number of tasks
  - Transport simulation is carried out independently by each task
  - Results are combined after the simulation is complete
- Advantages of this particular approach:
  - No communication between tasks until the end → almost linear scalability
  - Correlations between cycles are reduced → statistical errors may be more reliable?

Parallelization – MPI

- And the drawbacks:
  - Results are not shared during the simulation → single CPU calculation is not reproducable in parallel mode (complicates debugging)
  - Small population size per task may cause problems with statistics
  - No load sharing → calculation waits for the slowest task
  - *<u>Memory usage is multiplied by the number of parallel tasks</u>*
- Burnup and processing routines are parallelized by dividing the materials into separate tasks (completely independent calculations)

Parallelization – OpenMP

- Parallelization in Serpent 2 will be based on the combination of OpenMP and MPI
- The OpenMP part of the routines is already implemented:
  - Each parallel thread has the access to the same memory space
  - Parallelization takes place at the beginning of each neutron cycle – every neutron history is handled by its own thread
  - New random number generator
  - Parallelization of processing and burnup calculation similar to Serpent 1 (division by material)

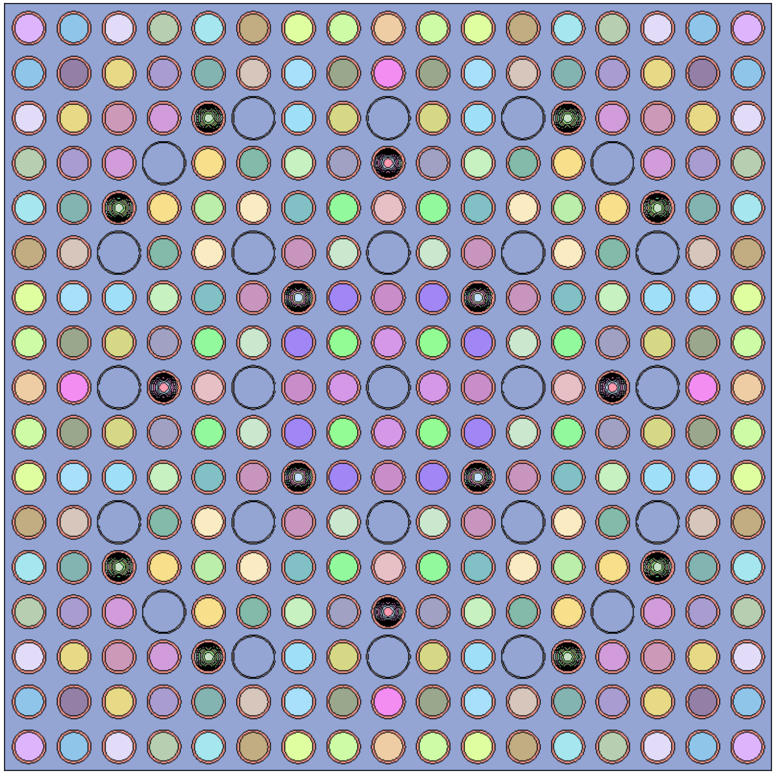


Parallelization – OpenMP

- Advantages of OpenMP:
  - Shared-memory technique – no extra storage space required
  - Relatively simple implementation, no data transfer
  - New RNG allows reproducability in parallel mode
- And the Drawbacks:
  - Writing in shared memory space requires run-time barriers or separate segments
  - Scalability is not very impressive and dependent on computer architecture (and possibly compiler?)

Parallelization – example

- The same 17 by 17 PWR assembly burnup calculation case
- Divided into 1 – 12 OpenMP threads
- Code compiled with gcc 4.1.2 (latest version is 4.6.1)
- Machine: 3.47 GHz Intel Xeon, 2 processors, 6 cores each
- Calculation run in optimization mode 4



Parallelization – example

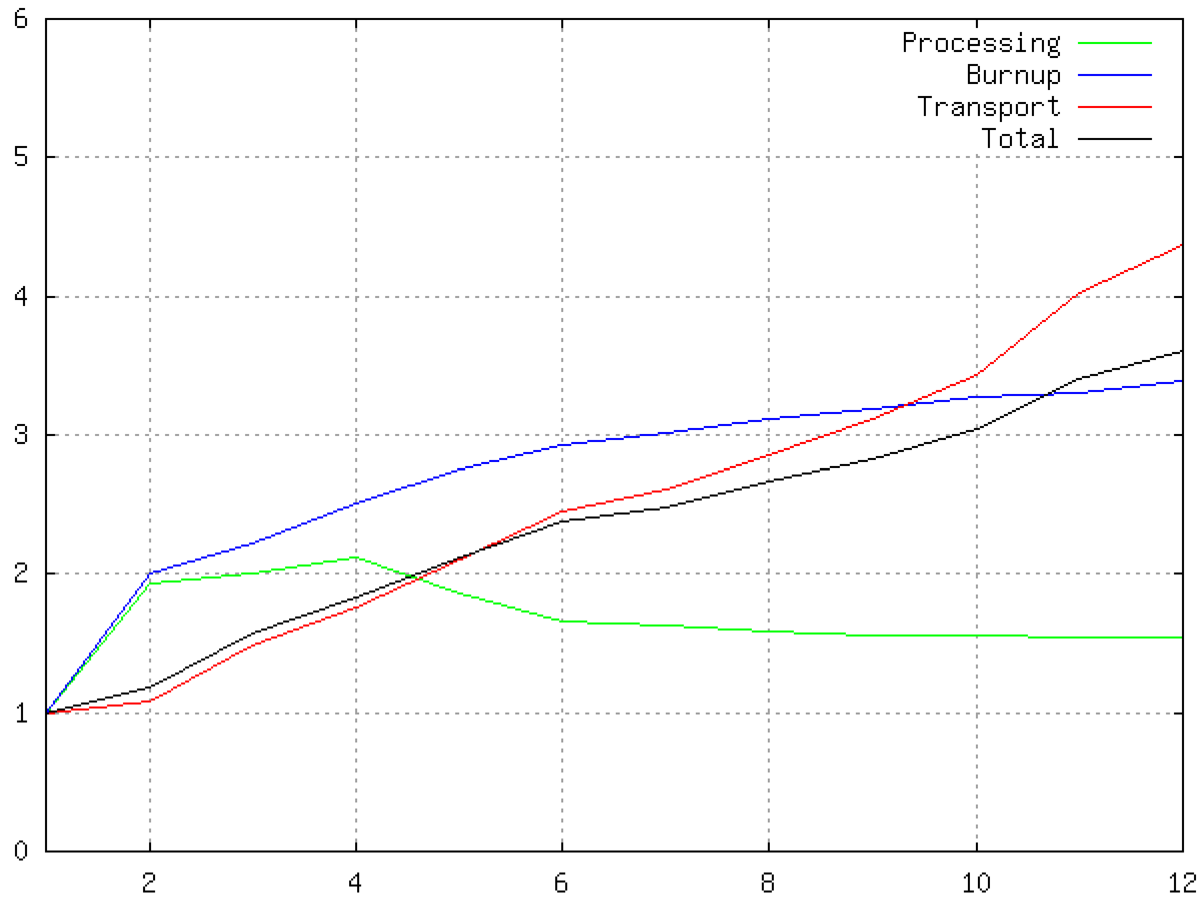


Fig 1. Speed-up factor as function of number of OpenMP threads

Parallelization – example

- The scalability of OpenMP parallelization is well below linear
- General observations:
  - Running the transport cycle requires setting barriers to prevent multiple threads from writing in the same memory space simultaneously – could this be the reason for poor scalability?
  - <u>However:</u> burnup and processing routines do not require any of these barriers and scalability is equally poor or worse?
- So is the poor scalability simply due to the nature of the calculation problem (constant memory access)?
- <u>But then again</u>: speed-up by a factor of 1.8 or more has been observed in some systems with 2 OpenMP threads!

New features in burnup calculation

- Apart from the memory issues, the methods used for burnup calculation in Serpent 1 do not require major revision.
- New features implemented and planned:
  - Secondary transmutation products (H, He-4, H-3) are included in the depletion chains
  - Energy-dependence of isomeric branching
  - Advanced time integration methods (another presentation)
  - Better options for depletion output
  - B1 criticality spectrum calculation to be extended in depletion
- CRAM routines are re-written and clearly superior to TTA, which will probably be left out from the final version

Photon physics

- One of the completely new features compared to Serpent 1 is the gamma transport simulation mode
- Independent mode already implemented with simplified physics (no production of secondary fluorescent or Brehmsstrahlung photons)
- To be added: source routine based on radioactive decay spectra, coupled neutron-gamma simulation, TTB approximation for secondaries
- Photon simulation has several similarities to neutron transport:
  - Neutral particles → linear transport problem
  - Transport routine similar to external source neutron simulation
  - Similar reaction types: absorption and two- and three-body scattering

Photon physics

- Differences to neutron transport:
  - Elemental, instead of isotopic reaction data
  - Smooth cross sections
  - Only four reaction modes: Thompson scattering, Compton scattering, photo-electric effect and pair production
  - No self-sustaining operation mode
- Photon transport seems to work well with delta-tracking and other techniques used in Serpent
- Most of the applications will probably be related to radiation shielding → variance reduction techniques will be required to improve statistics

Photon physics

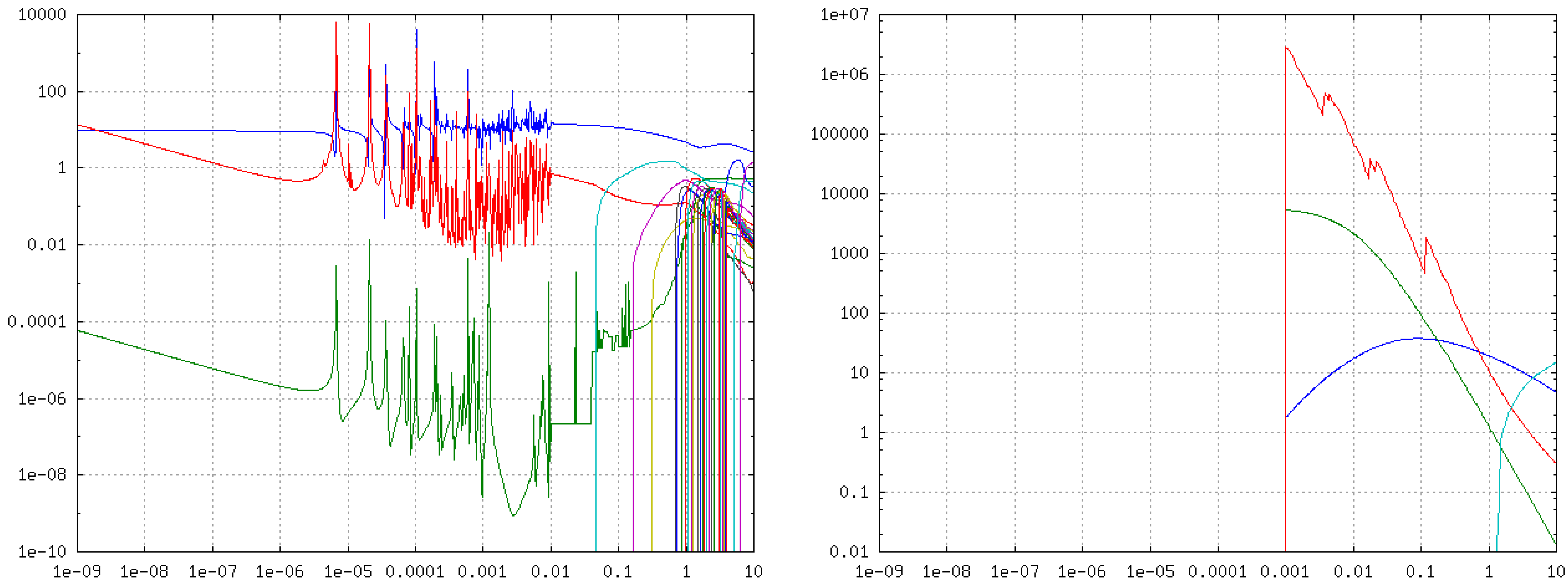


Fig 2. Left: neutron cross sections for U-238, Right: photon cross sections for uranium

Variance reduction techniques

- Serpent 1 is entirely based on *analog* Monte Carlo game:
  - Each simulated neutron history represents a single particle
  - Capture terminates neutron history
  - Multiplying (n,xn) scattering reactions divide history
  - Fission terminates history in criticality source simulation, and fission neutrons form the source for the next criticality cycle
  - Fission divides the history in external source simulation
- Analog Monte Carlo works well in Serpent because the code is mainly intended for reactor calculations, in which the *results are collected from the same region where the neutrons are born*

Variance reduction techniques

- Serpent 2 (like most Monte Carlo codes) will have several options for *implicit* Monte Carlo game:
  - Each neutron (or photon) history is associated with a statistical weight
  - Implicit capture reduces the weight according to capture probability (history is not terminated)
  - Implicit (n,xn) and fission multiply the weight
- The idea of implicit techniques is to get more particles in regions where they are not willing to go → better statistics (especially in shielding calculations)
- The particle weight is adjusted to compensate for the bias introduced from cheating in the game

Variance reduction techniques

- What has been done so far:
  - Particle weight is a variable similar to position and energy, and it is carried through the simulation
  - Implicit (n,xn) is used by default, and the results seem OK
  - Implicit capture is optional, but not used by default (may have some compatibility issues with other calculation methods)
  - Implicit fission is a curiosity that may not be included in the final version
  - Some testing has been carried out with basic variance reduction techniques (splitting, Russian roulette, etc.)
- Advanced variance reduction techniques (weight windows, etc.) will be a major topic for future studies

Variance reduction techniques – example

- The capability to adjust the number of neutrons during the simulation has allowed the implementation of a simple method, denoted here as source biasing (not sure about the terminology):

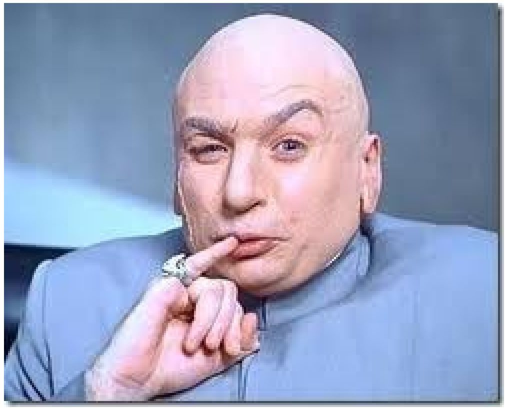
  - The geometry is covered by a three-dimensional Cartesian mesh
  - The number of fission neutrons emitted in each mesh cell is counted as the calculation proceeds
  - The number of fission neutrons and neutron weight for every source point is adjusted according to the fraction of previously recorded source points in the mesh cell
  - The main goal is to get a *uniform distribution* over the entire source region and better statistics in large (full core) geometries
  - NOTE: this is more about playing around with neutron weights and splitting, the theoretical basis for the method has not been verified

Variance reduction techniques – example

- The source biasing method was tested using the Hoogenboom-Martin Monte Carlo performance benchmark:
  - Full-scale PWR core geometry with simplified material compositions
  - Calculation of core power distribution at pin level, with each pin divided into 100 axial segments → over 6 million tally regions
  - Main goal is to get the relative statistical errors < 1% in all regions
- The benchmark was set up in order to follow the development of computer capacity and Monte Carlo codes, and the possibility of using the continuous-energy Monte Carlo method for TH-coupled full-core calculations

Variance reduction techniques – example

- The benchmark was calculated earlier with Serpent 1.1.13:[1]
  - 100 billion (100,000,000,000) neutron histories run
  - 5 months of CPU time
  - Target accuracy of 1% reached in 60% of the regions



- Similar calculation with Super:
  - 20 billion (20,000,000,000) neutron histories run with and without source biasing
  - Calculations are still running (about ¾ complete)

[1)] J. Leppänen. *"Use of the Serpent Monte Carlo Reactor Physics Code for Full-Core Calculations"* In proc. SNA + MC2010, Tokyo, Japan, October 17-21, 2010.

Variance reduction techniques – example

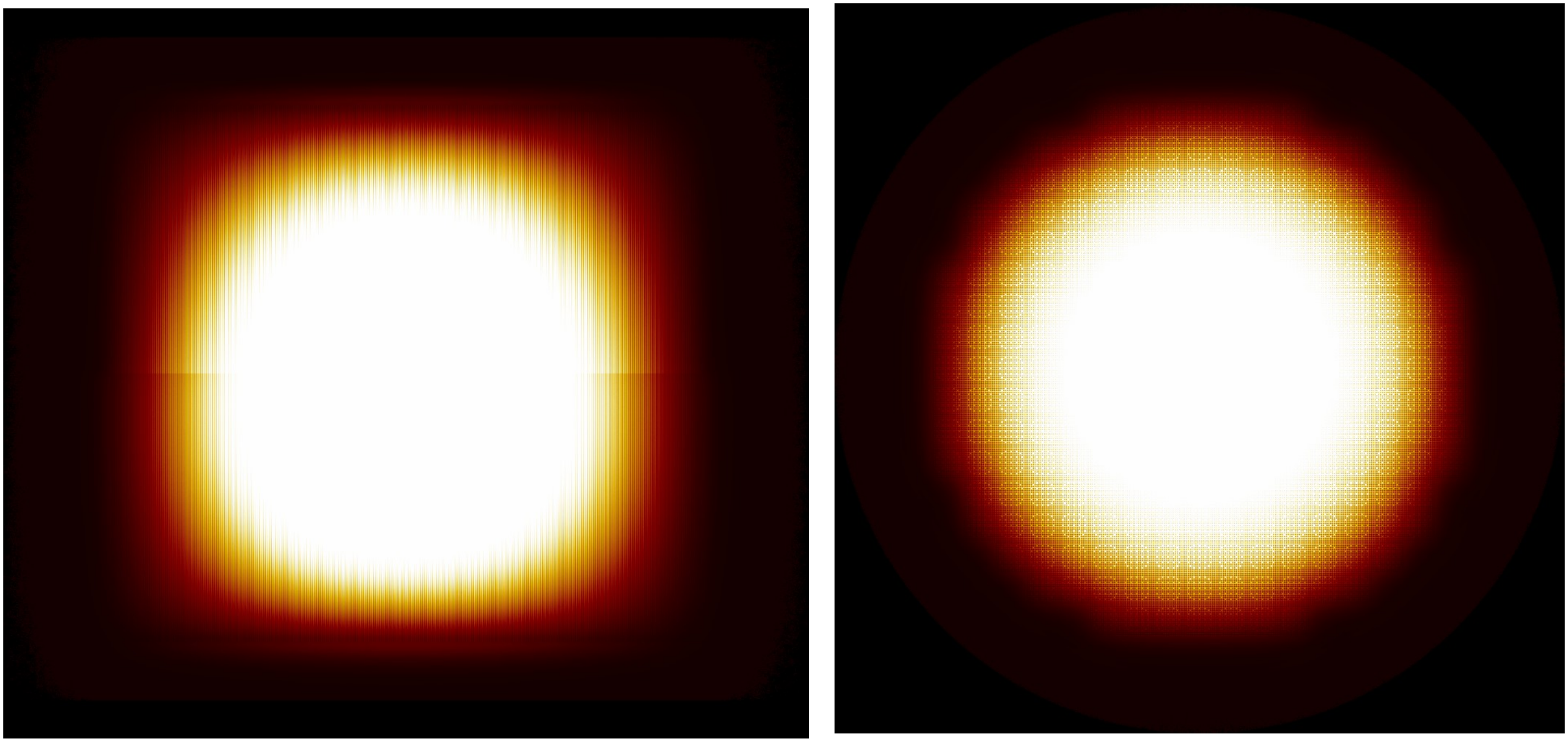


Fig 3. Source distribution without source biasing method (left: side view, right: top view)

Variance reduction techniques – example

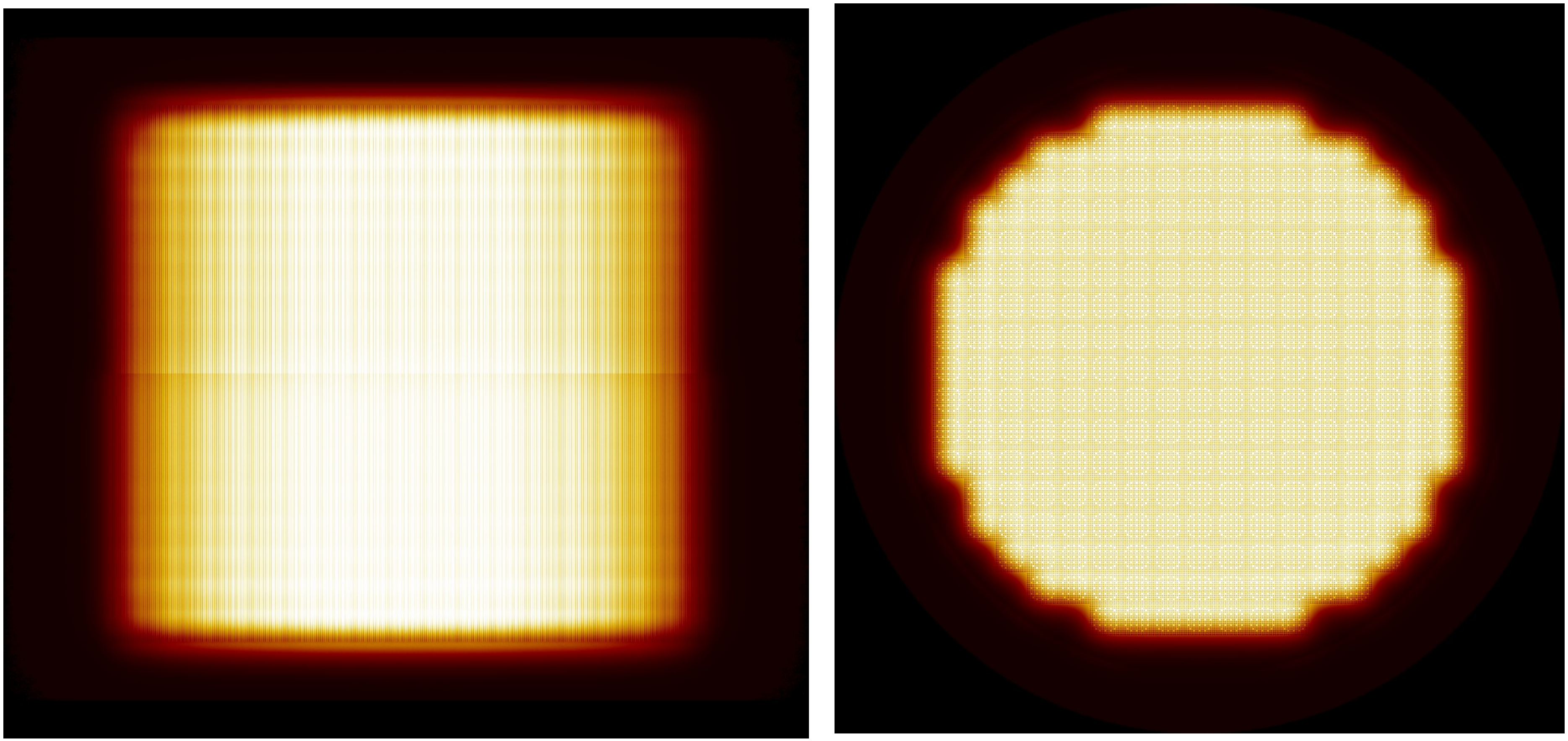
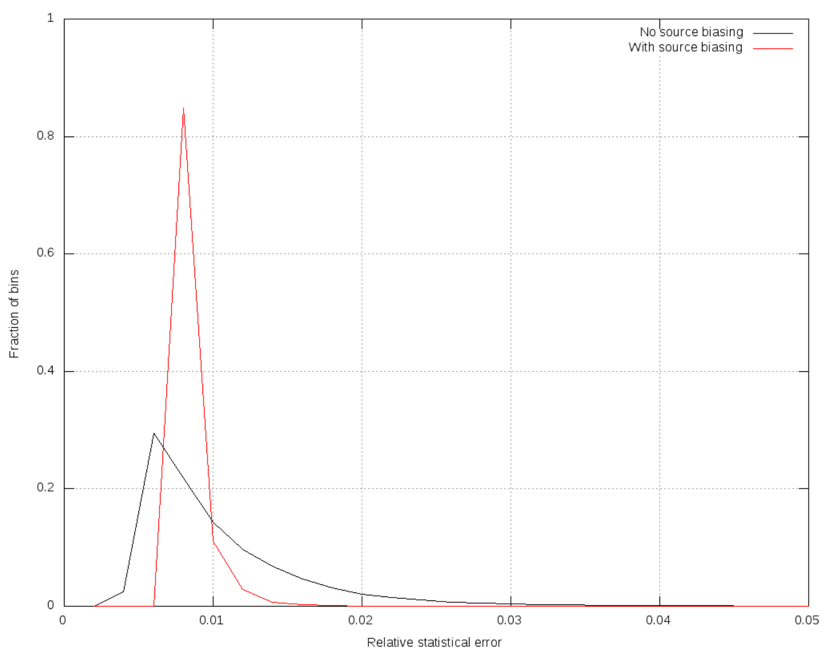


Fig 4. Source distribution with source biasing method (left: side view, right: top view)

Variance reduction techniques – example



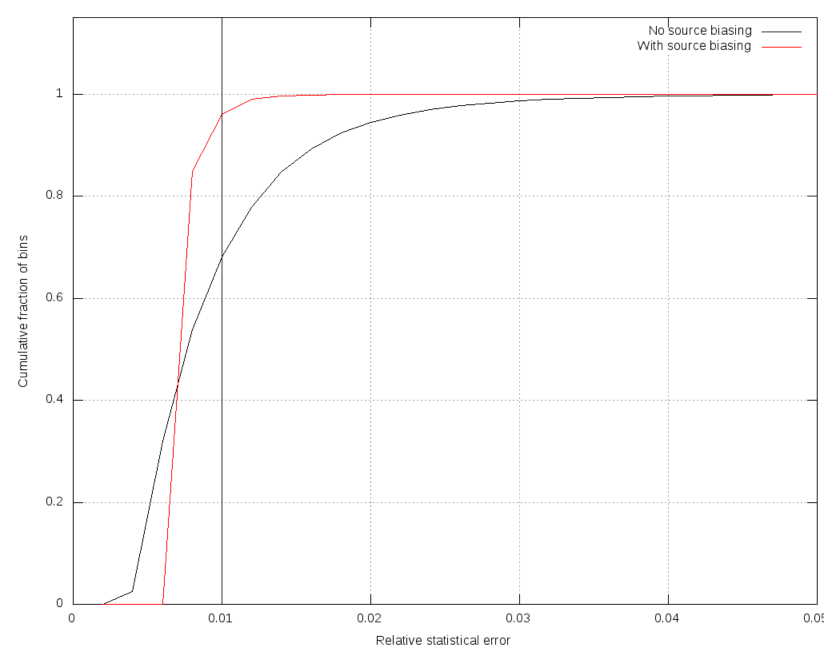
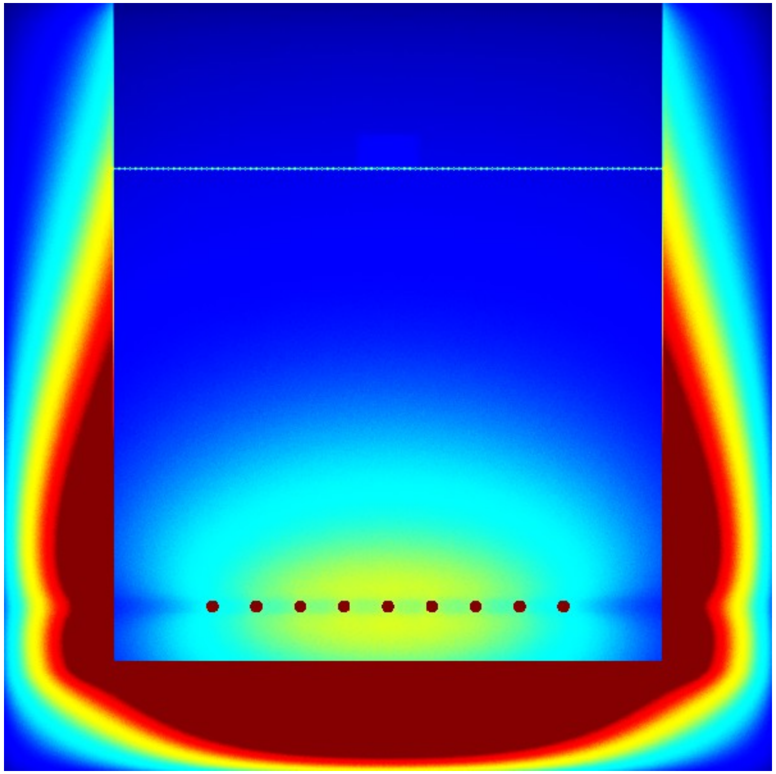


Fig 5. Left: error distribution in the 6 million regions, Right: cumulative distribution functions

Additional new features compared to Serpent 1

- Union operator for constructing cells (easier conversion between Serpent and MCNP geometry formats)
- More options for nest-type geometries (nests not limited to a single surface type)
- Material mixtures (mass or volumetric mixing of one or more materials)
- New options for mesh plots (color maps, collision, gamma heat, etc. distributions and visualization of detector response functions)



What's next?

- Current status of Serpent 2:
  - Neutron physics and basic features (geometry routine, group constant generation, detectors, burnup calculation) are more or less completed
  - Parallelization works with OpenMP, but the performance should be better
  - Development of photon transport routines has been started
  - Everything should be ready for the implementation of advanced variance reduction techniques

  (The last two will require some studying in my part)

What's next?

- Next in the to-do list:
  - Unresolved resonance probability table treatment must be verified and optimized
  - Implementation of MPI parallelization
  - Important extra features from Serpent 1: equilibrium xenon calculation, DBRC, critical spectrum calculation
- Once these capabilities are implemented (hopefully by the end of the year), the code is ready to be released for beta-testing:
  - Distribution to existing users with time, interest and patience
  - No public NEA / RSICC distribution at this stage (maybe mid 2012?)

What's next?

- Challenges for code validation:
  - A lot of options and combinations to be tested: optimization modes, implicit reactions, unresolved resonance probability table sampling, parallelization with OpenMP and MPI
  - Very large burnup calculation problems (> 1000 burnable materials) may bring new challenges for methods and optimization
  - Entirely new features: gamma and coupled neutron-gamma transport, variance reduction techniques

What's next?

- Hot topics and future plans:
  - An on-the-fly Doppler broadening routine is currently under development (another presentation)
  - Adjoint Monte Carlo calculation using the iterated fission probability (IFP) method:
    - Calculation of adjoint-weighed kinetic parameters
    - Perturbations
    - *Huge* potential for variance reduction
    - Applications in sensitivity and uncertainty analysis

What's next?

- Multi-physics:
  - Coupling to thermal-hydraulics codes (Serpent-PORFLO coupling in the framework of the EU HPMC project)
  - Coupling of Serpent and fuel performance codes (some Serpent-ENIGMA calculations already done)
  - <u>*Development of a general-purpose interface for the exchange of input and output data between codes*</u>

That's it – thank you for listening!

Questions?

Jaakko.Leppanen@vtt.fi

Serpent 2 -related discussion area at the Serpent forum: http://ttuki.vtt.fi/serpent/

More info coming sooner or later at the website: http://montecarlo.vtt.fi/