

Multithreading I

SLAC Geant4 Tutorial 2014

Monday March 3 2014 - Jen-Hsun Huang Engineering Center Stanford University

A. Dotti (adotti@slac.stanford.edu)

Important Note

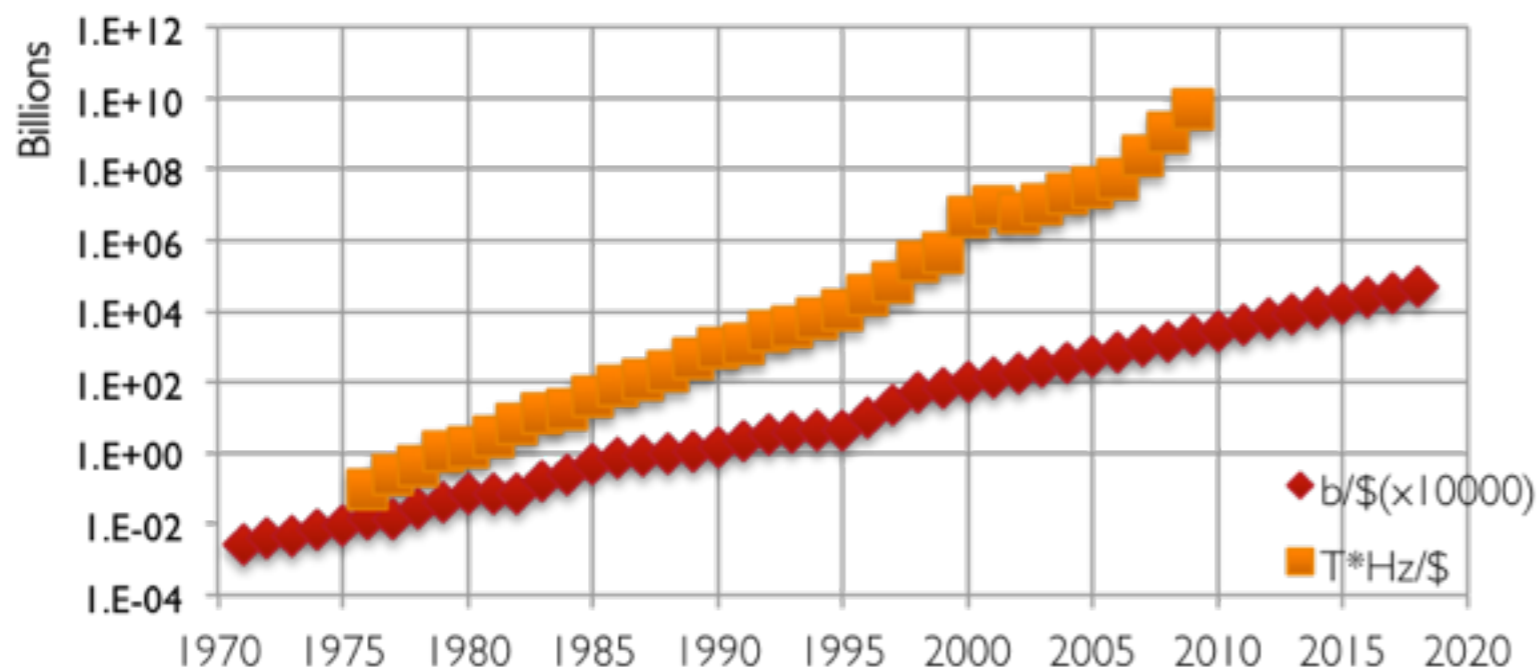
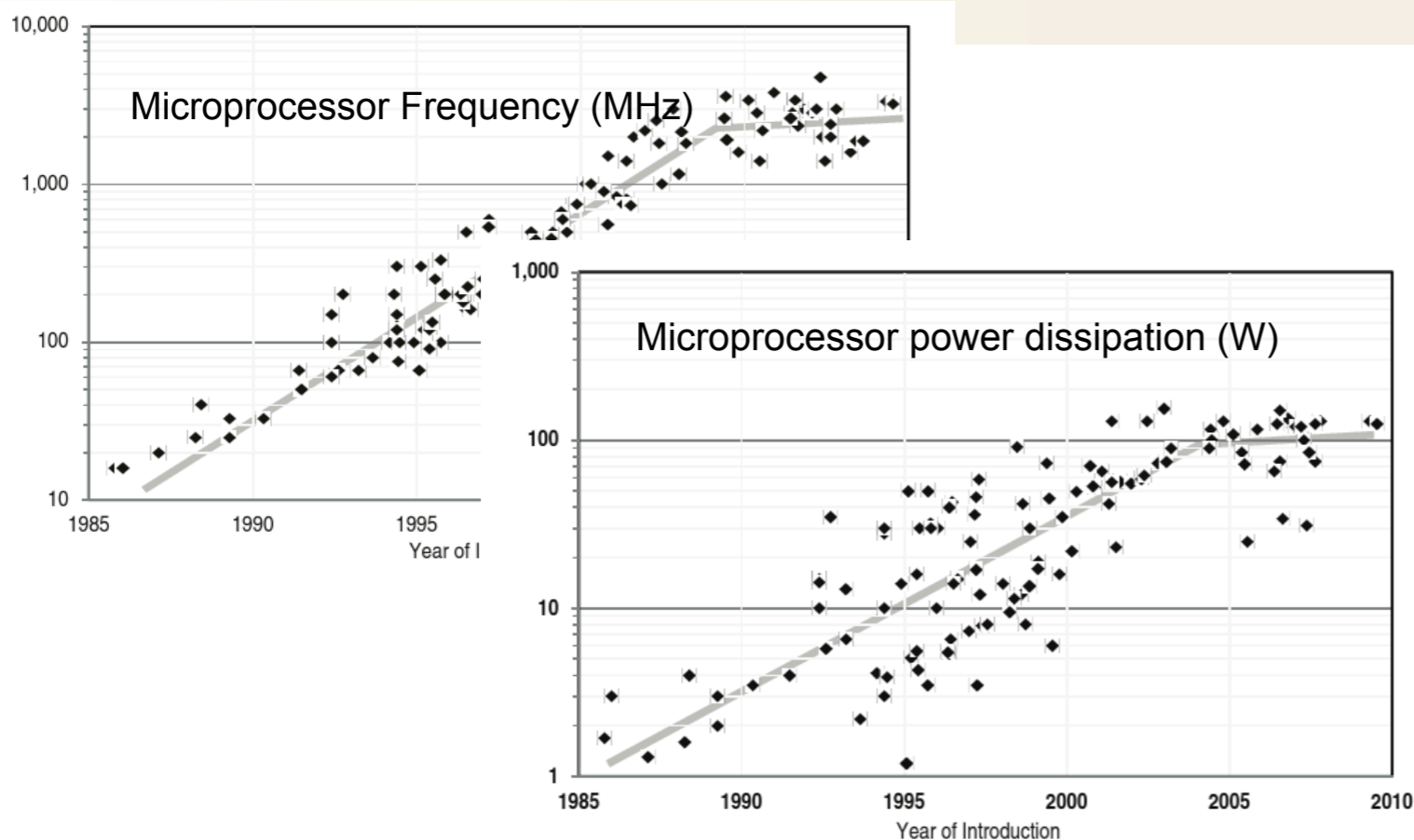


This is the first part of a 2- session

Only the **very minimum** will be introduced here

- What is multi-threading?
- How to activate MT
- How to migrate code (thread-safety in second talk)
- UI commands related to MT

The challenges of many-core era



- Increase frequency of CPU causes **increase of power needs**
- Reached plateau around 2005
 - No more increase in CPU frequency
- However number of transistors /\$ you can buy continues to grow
 - Multi/May-core era
- Note: quantity memory you can buy with same \$ scales slower
- **Expect:**
 - Many core (double/2yrs?)
 - Single core performance will not increase as we were used to
 - Less memory/core
- New software models need to take these into account: increase parallelism

CPU Clock Frequency land usage: The Future of Computing Performance: Game Over or Next Level?

DRAM cost: Data from 1971-2000: VLSI Research Inc. Data from 2001-2002: ITRS, 2002 Update, Table 7a, Cost-Near-Term Years, p. 172. Data from 2003-2018: ITRS, 2004 Update, Tables 7a and 7b, Cost-Near-Term Years, pp. 20-21.

CPU cost: Data from 1976-1999: E. R. Berndt, E. R. Dulberger, and N. J. Rappaport, "Price and Quality of Desktop and Mobile Personal Computers: A Quarter Century of History," July 17, 2000. ;Data from 2001-2016: ITRS, 2002 Update, On-Chip Local Clock in Table 4c: Performance and Package Chips: Frequency On-Chip Wiring Levels -- Near-Term Years, p. 167. ; Average transistor price: Intel and Dataquest reports (December 2002), see Gordon E. Moore, "Our Revolution,"

In Brief



- Modern CPU architectures: need to introduce **parallelism**
- Memory and its access will limit number of concurrent processes running on single chip
- Solution: add parallelism **in the application code**

- Geant4 needs back-compatibility with user code and **simple approach** (physicists \neq computer scientists)
- **Events are independent**: each event can be simulated separately
- Multi-threading for event level parallelism is the natural choice

Geant4 Multi Threading capabilities

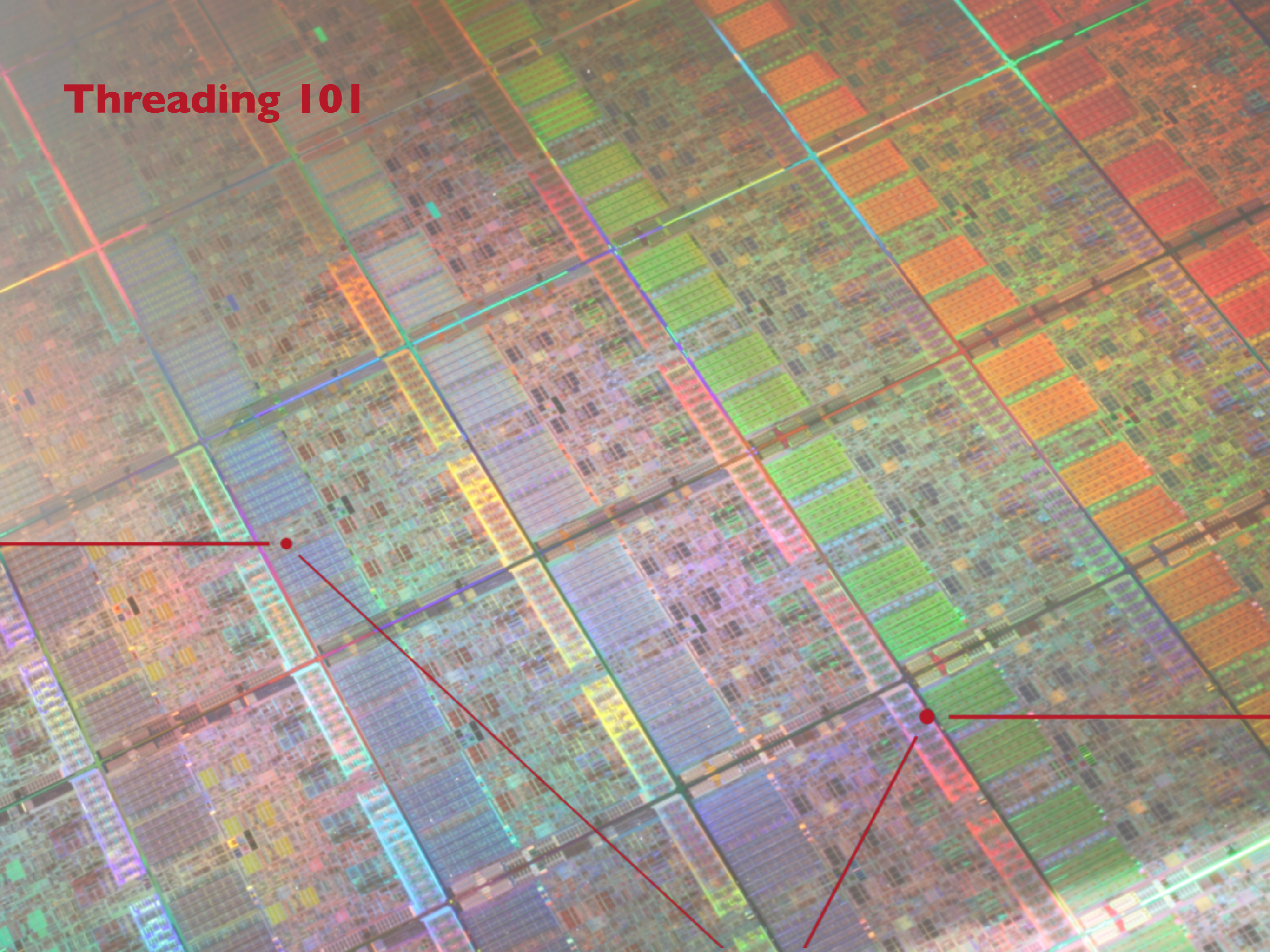


- MT code integrated into G4
- Public release
- All functionalities ported to MT



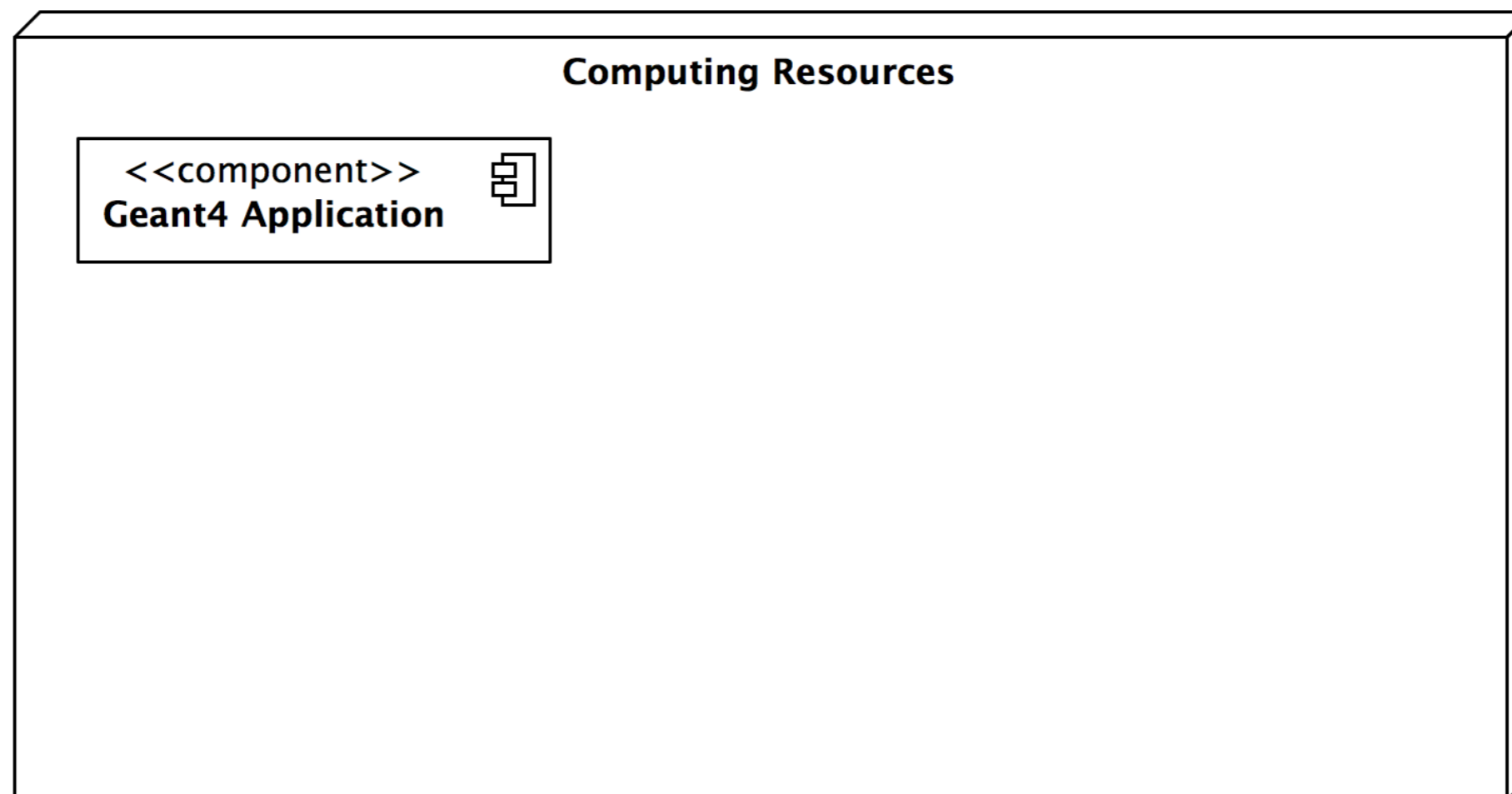
- Proof of principle
- Identify objects to be shared
- First testing
- API re-design
- Example migration
- Further testing
- First optimizations
- Further Refinements
- Focus on further performance improvements

Threading IOI



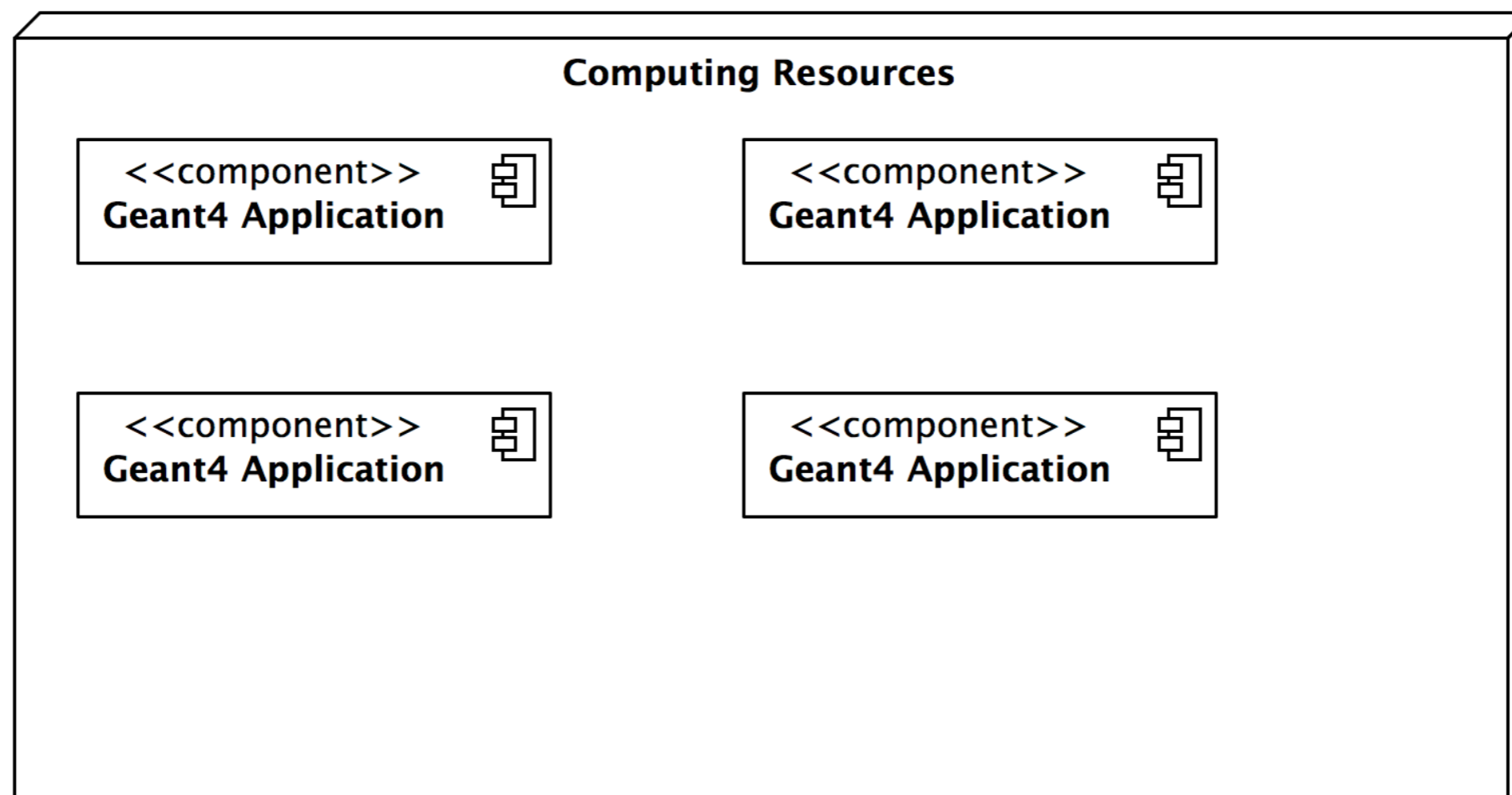
What is a thread?

Sequential application



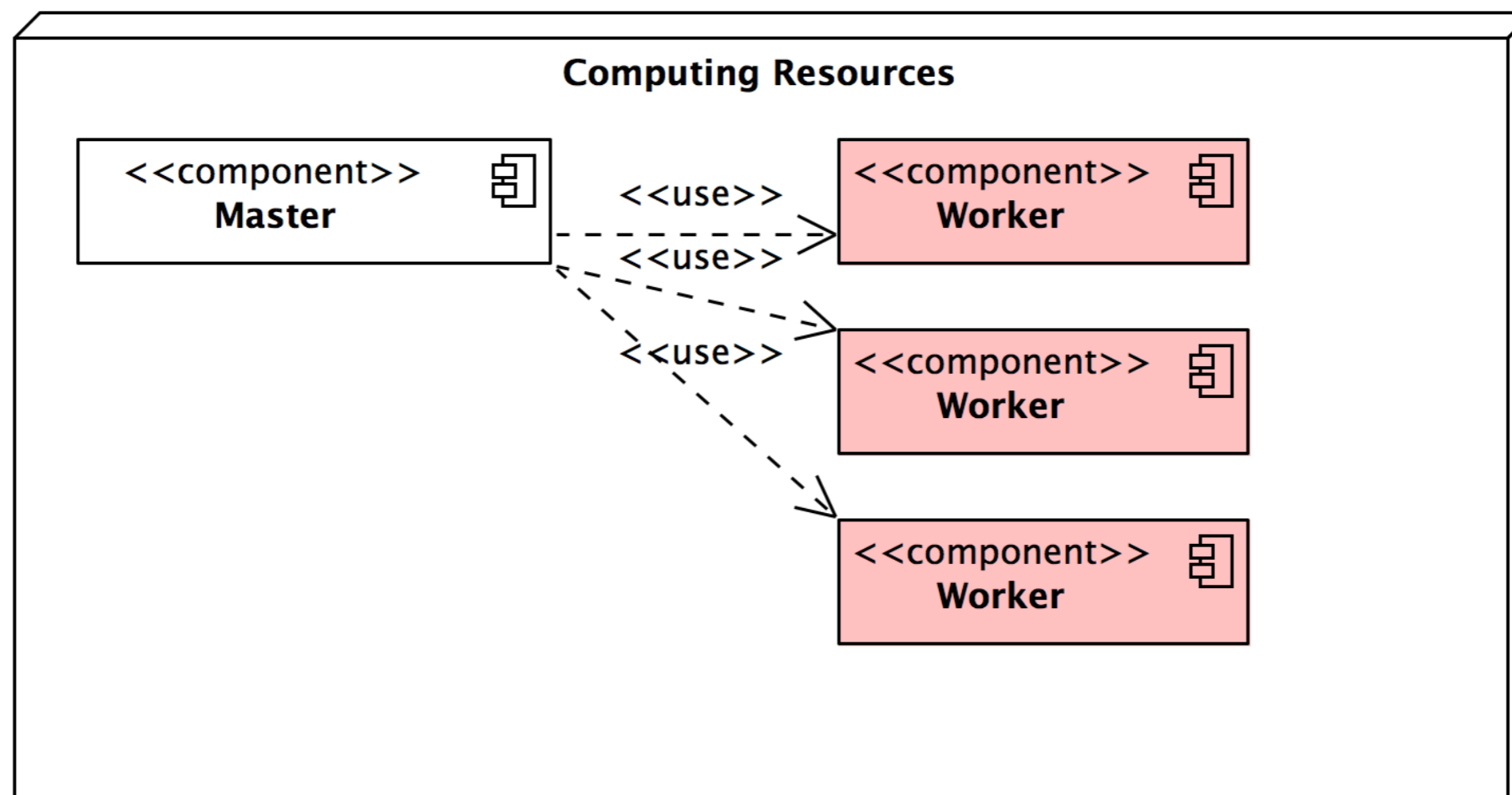
What is a thread?

Sequential application: start N (cores/CPU) copies of application if fits in memory



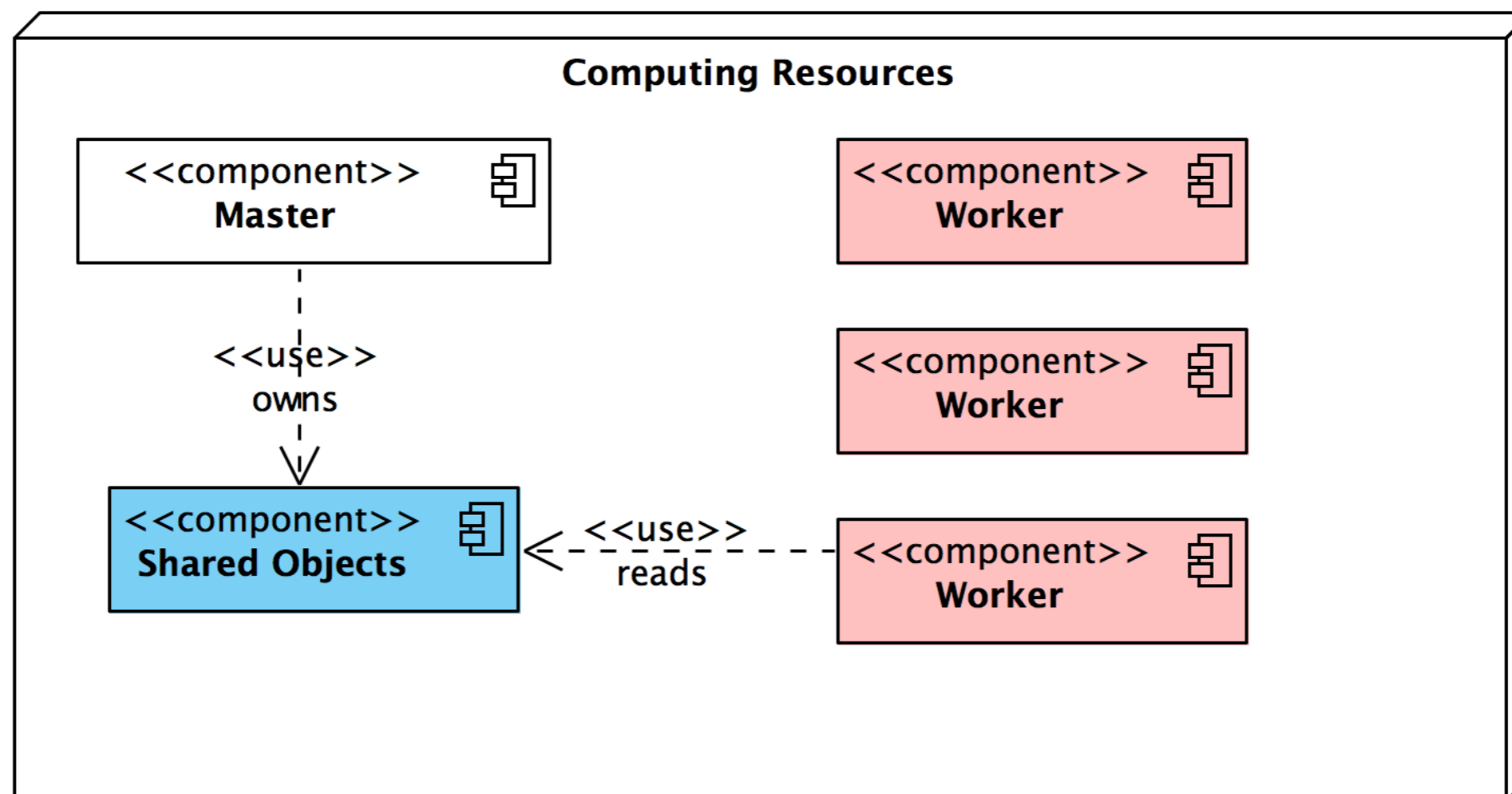
What is a thread?

MT Application: single application starts threads. For G4: application (master) controls workers that do simulation, no memory sharing now, each worker is a copy of the application

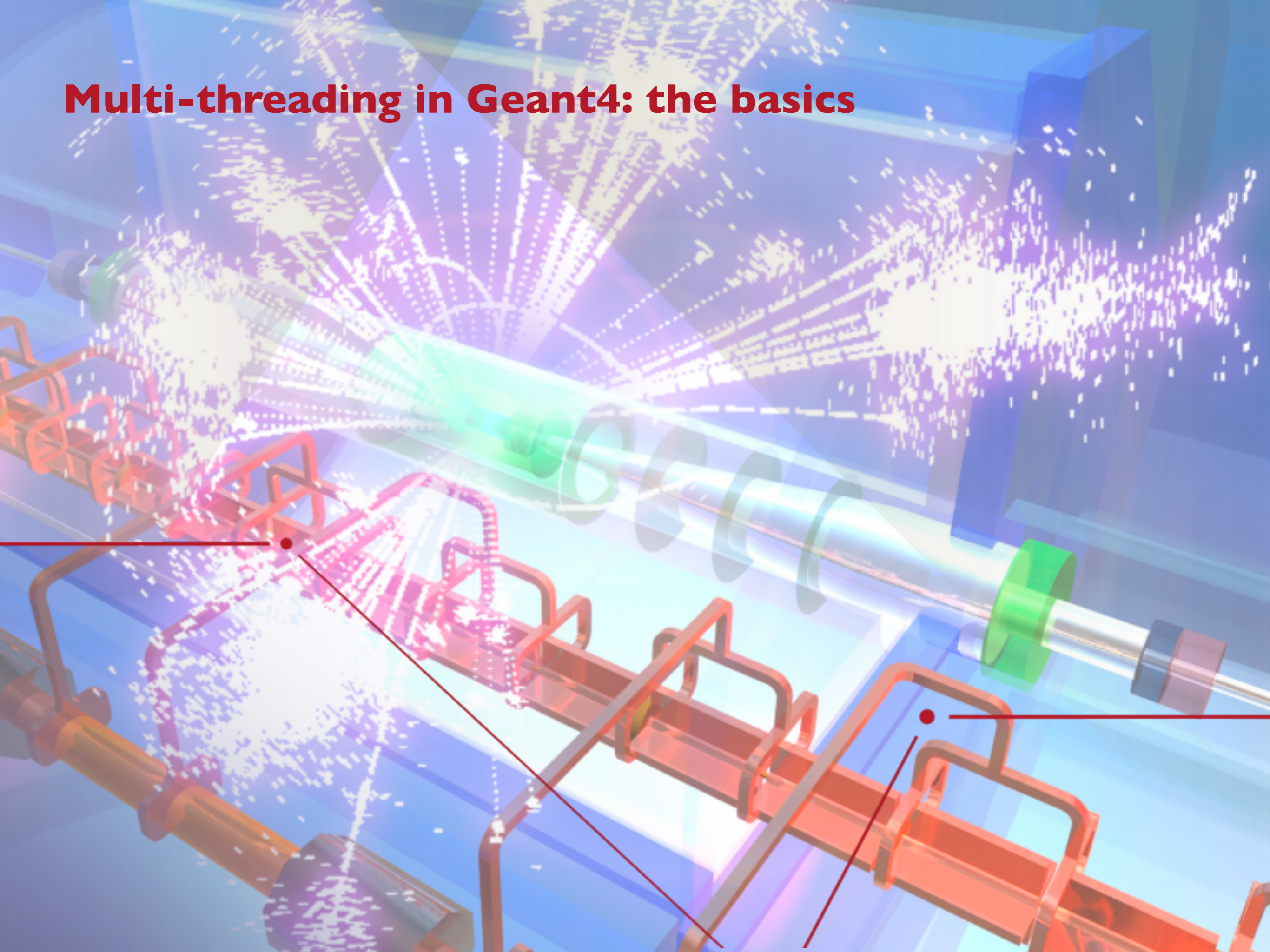


What is a thread?

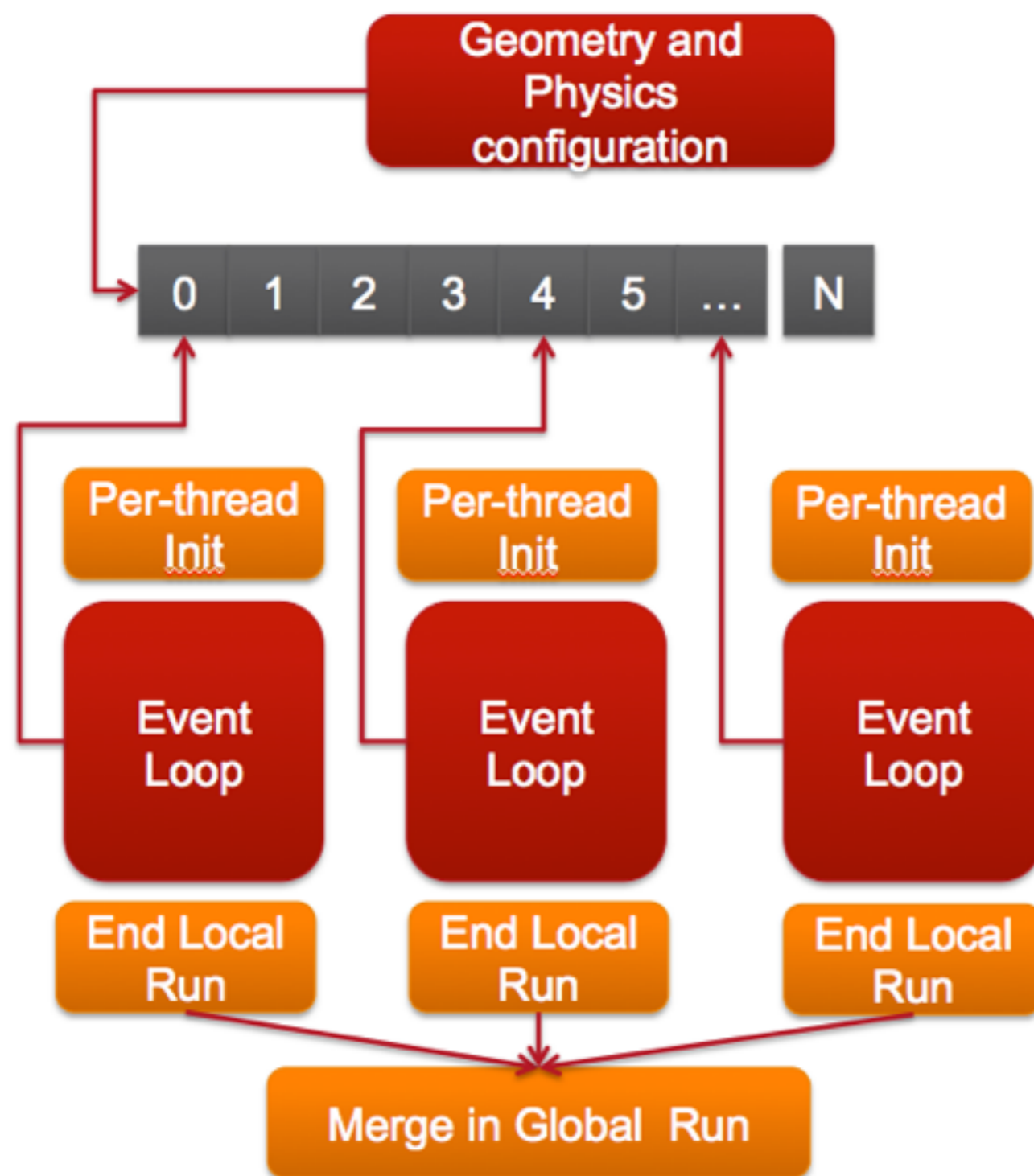
Memory reduction: introduce shared objects, memory of N threads is less than memory used by N copies of application



Multi-threading in Geant4: the basics



General Design



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

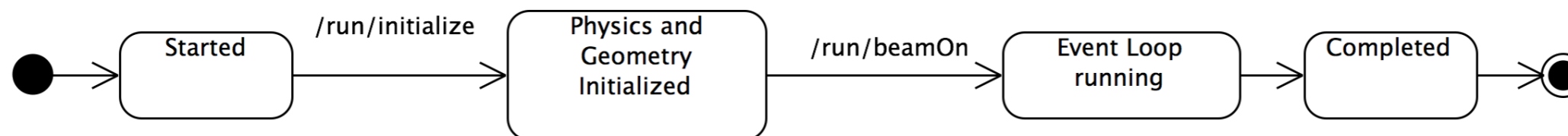
Threads compete for next event to be processed

Command line scoring and G4tools automatically merge results from threads

Simplified Master / Worker Model



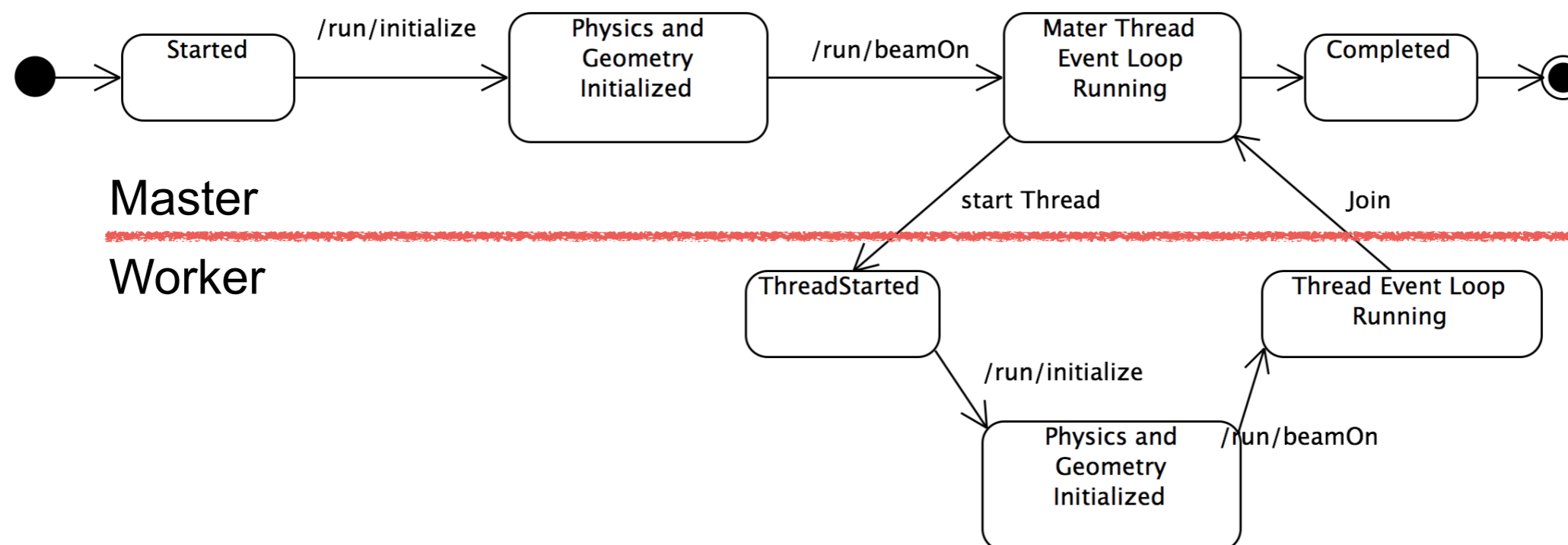
- A G4 (with MT) application can be seen as simple finite state machine



Simplified Master / Worker Model



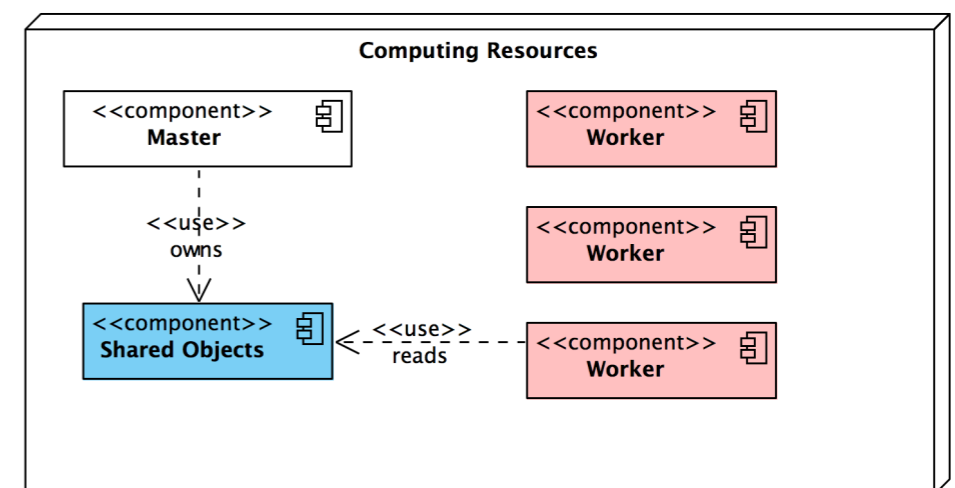
- A G4 (with MT) application can be seen as simple finite state machine
- Threads do not exist before first /run/beamOn
- When master starts the first run spawns threads and distribute work



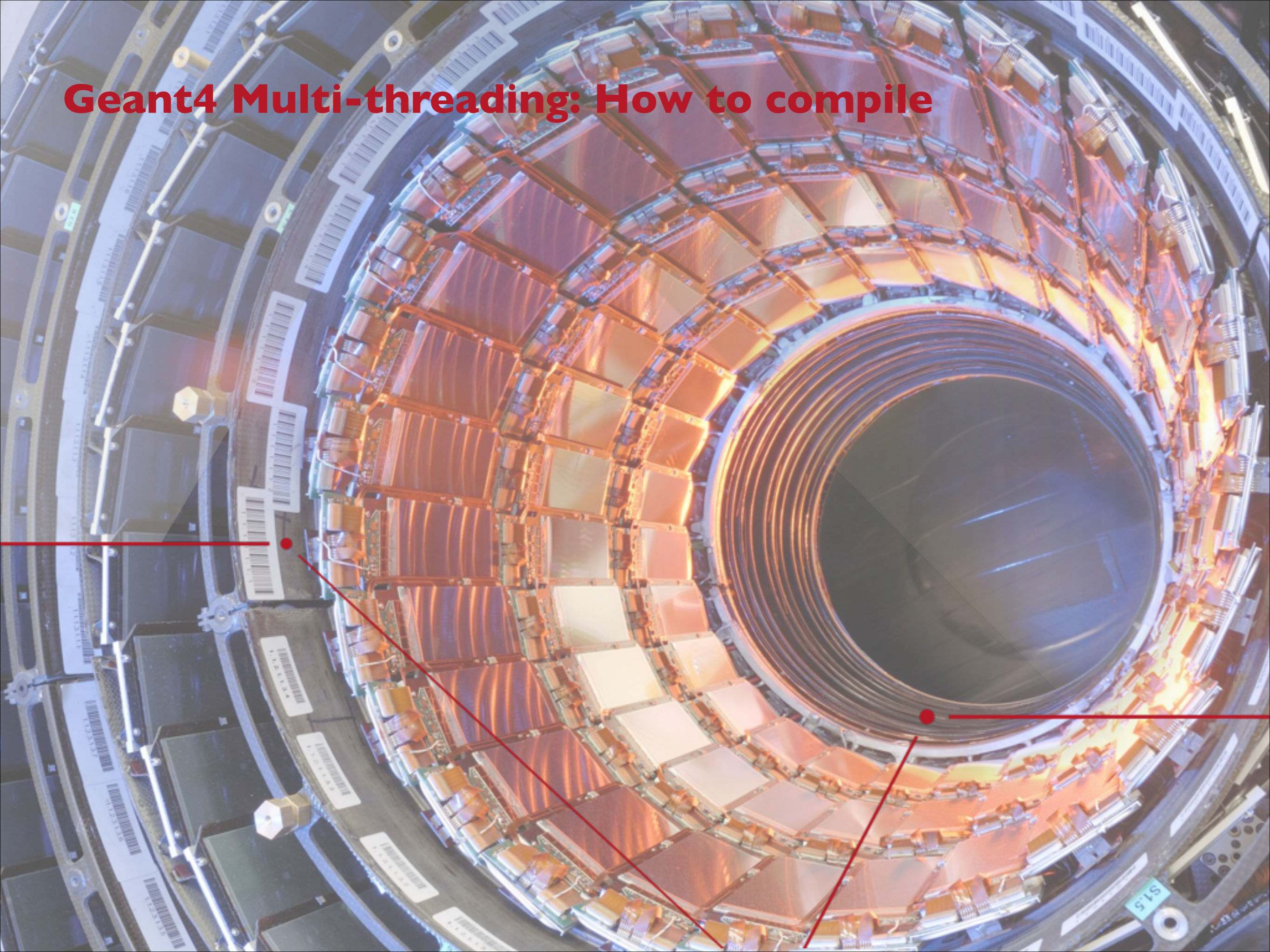
Shared Vs Thread-local



- To reduce memory footprint threads must share at least part of the objects
- **General rule in G4: threads can share whatever is invariant during the event loop** (e.g. threads do not change these objects while processing events, these are used “read-only”)
 - Geometry definition
 - Electromagnetic physics tables
 - The reason for this is discussed in second part



Geant4 Multi-threading: How to compile



How to configure Geant4 for MT



- `cmake -DGEANT4_BUILD_MULTITHREADED=ON [...]`
- Requires “recent” compiler that supports ThreadLocalStorage technology (to be discussed Thursday) and pthread library installed (usually pre-installed on POSIX systems)
- Check cmake output for:
 - Performing Test HAVE_TLS
 - Performing Test HAVE_TLS - Success
- If it complains then your compiler is too old, sorry...
- Mac OS X, you need to use clang \geq 3.0 (not gcc!). On Mac OS X 10.7:
`cmake -DCMAKE_CXX_COMPILER=clang++ -DCMAKE_C_COMPILER=clang \
-DGEANT4_BUILD_MULTITHREADED=ON [...]`
- Sorry no WIN support!
- Compile as usual

Code Compatibility



- Some API have changed to enable MT (this is why this is a major release)
 - The exercises of this tutorial will show how to implement these correctly for MT
- **You can use an application developed for G4 Ver 9.6 without changing your code in sequential mode** (except for other mandatory modifications not MT-related)
- **An MT-ready application, can also run in sequential mode without changing your code** (but not vice-versa)

Application Changes: How to 1/2



- **Detector Construction** two functions to implement:
 - `G4VPhysicalVolume* G4VUserDetectorConstruction::Construct();`
 - Build here your detector geometry except Sensitive Detectors and magnetic field (called by master thread once)
 - `void G4VUserDetectorConstruction::ConstructSDandField();`
 - Build here SDs and B-Fields (called by each thread)
- **Create a new class** that inherits from `G4VUserActionInitialization` and implement:
 - `void G4VUserActionInitialization::Build()`
 - Instantiate here user-actions for worker threads (called by each thread)
 - `void G4VUserActionInitialization::BuildForMaster()`
 - Instantiate here user-actions for master (optional) (called by master)

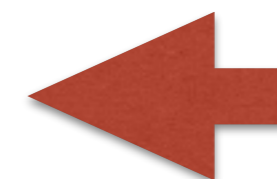
If you are new to Geant4 this slide will become clear during the tutorial: you may want to review this presentation later in the week

Application Changes: How to 2/2



- In the main() function instantiate a G4MTRunManager and (if you want) set the default number of threads:

```
int main(int,char**) {  
    #ifdef G4MULTITHREADED  
        G4MTRunManager* runManager = new G4MTRunManager;  
        runManager->SetNumberOfThreads(G4Threading::G4GetNumberOfCores());  
    #else  
        G4RunManager* runManager = new G4RunManager;  
    #endif  
  
    // Mandatory user initialization classes  
    runManager->SetUserInitialization(new DetectorConstruction);  
  
    G4VModularPhysicsList* physicsList = new FTFP_BERT;  
    runManager->SetUserInitialization(physicsList);  
  
    // User action initialization  
    runManager->SetUserInitialization(new ActionInitialization());  
    //...
```



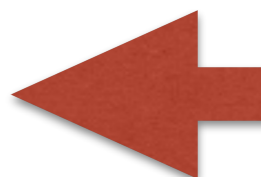
Compiled if MT=ON

Application Changes: How to 2/2



- In the main() function instantiate a G4MTRunManager and (if you want) set the default number of threads:

```
int main(int, char**) {  
    #ifdef G4MULTITHREADED  
        G4MTRunManager* runManager = new G4MTRunManager;  
        runManager->SetNumberOfThreads(G4Threading::G4GetNumberOfCores());  
    #else  
        G4RunManager* runManager = new G4RunManager;  
    #endif  
  
    // Mandatory user initialization classes  
    runManager->SetUserInitialization(new DetectorConstruction);  
  
    G4VModularPhysicsList* physicsList = new FTFP_BERT;  
    runManager->SetUserInitialization(physicsList);  
  
    // User action initialization  
    runManager->SetUserInitialization(new ActionInitialization());  
    //...
```



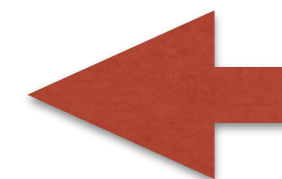
Returns number of
logical cores of machine

Application Changes: How to 2/2



- In the main() function instantiate a G4MTRunManager and (if you want) set the default number of threads:

```
int main(int, char**) {  
    #ifdef G4MULTITHREADED  
        G4MTRunManager* runManager = new G4MTRunManager;  
        runManager->SetNumberOfThreads(G4Threading::G4GetNumberOfCores());  
    #else  
        G4RunManager* runManager = new G4RunManager;  
    #endif  
  
    // Mandatory user initialization classes  
    runManager->SetUserInitialization(new DetectorConstruction);  
  
    G4VModularPhysicsList* physicsList = new FTFP_BERT;  
    runManager->SetUserInitialization(physicsList);  
  
    // User action initialization  
    runManager->SetUserInitialization(new ActionInitialization());  
    //...
```



Compiled if MT=OFF

UI Commands



SLAC National Accelerator Laboratory

Photo Credit: Peter Ginter

MT related UI commands



/run/numberOfThreads [n] : Specify number of threads

/control/cout/setCoutFile [filename] : Sends G4cout stream to a per-thread file. Use "***Screen***" to reset to screen

/control/cout/setCerrFile [filename] : As previous but for G4cerr

/control/cout/useBuffer [true|false] : Send G4cout/G4cerr to a per-thread buffer that will be printed at the end of the job

/control/cout/prefixString [string] : Add a per-thread identifier to each output line from threads, the thread id is appended to this prefix (default: G4WTn)

/control/cout/ignoreThreadsExcept [id] : Show output only from thread "id"

Setting the number of threads



- Default the number of threads: 2
 - Use `/run/numberOfThreads` or `G4MTRunManager::SetNumberOfThreads()`
- `G4Threading::G4GetNumberOfCores()` returns the number of logical cores of your machine
- Currently number of threads **cannot be changed** after `/run/initialize` (C++ call to: `G4RunManager::Initialize()`)
- You can overwrite your application behavior and UI commands setting the (shell) environment variables `G4FORCENUMBEROFTHREADS=...` before starting the application (the special keyword **max** can be used to use all system cores)

User Defined UI commands



- User interacts with application typing UI commands
 - Master thread “accumulates” the commands and passes the commands stack to all the threads at the beginning of a run
 - Threads execute the same commands sequence as master thread
- However some commands **make sense only in master thread** (e.g. the one modifying the geometry)
- UI commands can be marked as “not to be broadcasted”:
 - **`G4UIcommand::SetToBeBroadcasted(false);`**
- Do not forget this step if you implement user-defined UI commands

Conclusions



Parallelism is a tricky business:

- User code has to be thread-safe
- Race conditions may appear (better: they will very probably appear)
- Bugs may often seem “random” and difficult to reproduce
- Experience is needed for complex applications, but we believe for simple ones following these instructions is enough
- A new hyper news user forum has been created (Multithreading) to address all possible questions
- Ask an expert!