

# Migration of Examples and User Application Code to MT

I. Hrivnacova, IPN Orsay  
P. Gumplinger, TRIUMF

18<sup>th</sup> Geant4 Collaboration Meeting,  
23 - 27 September 2012, Seville

# Outline

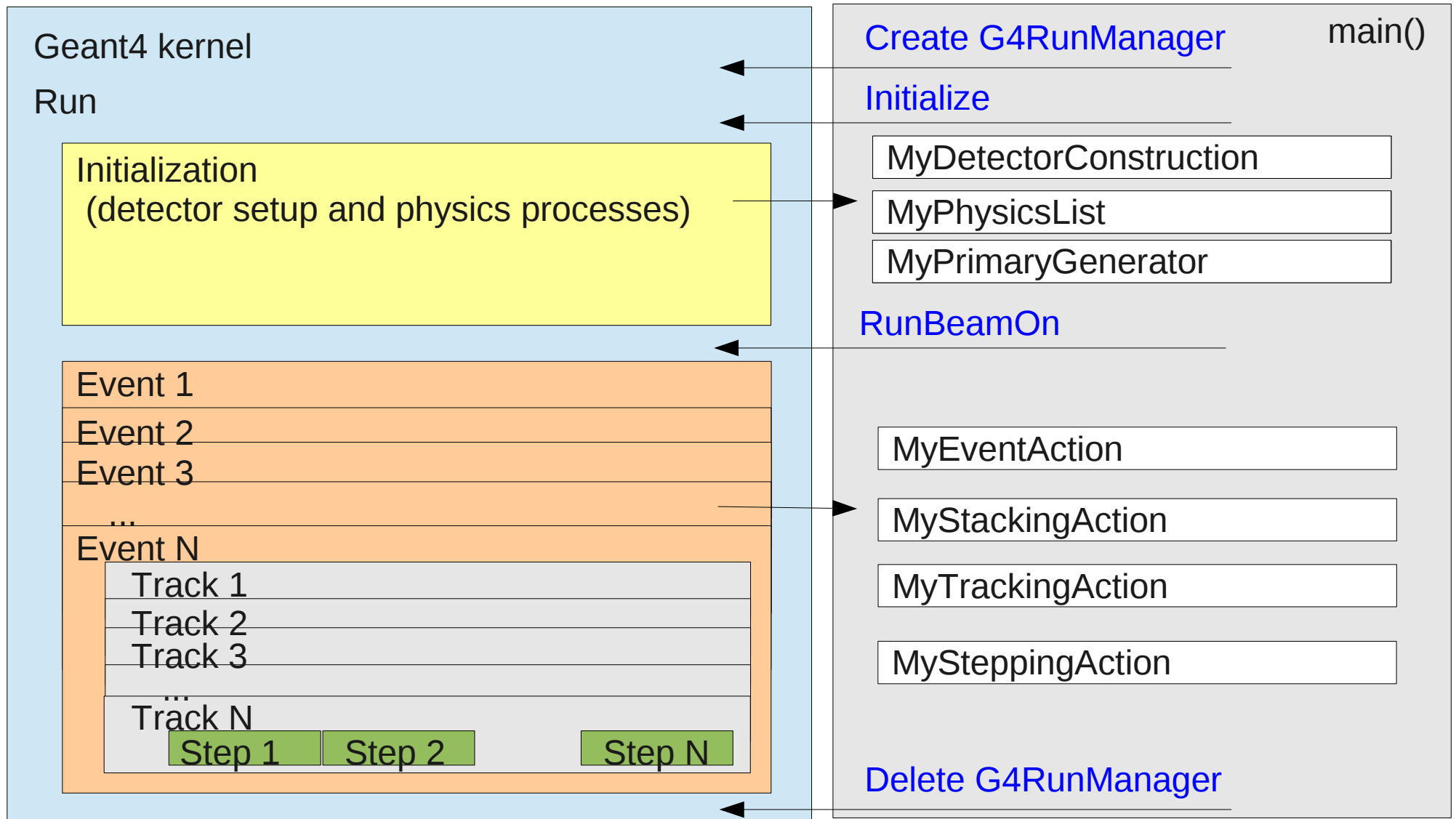
- Where to start
- Basic understanding of MT mode
- MT migration of a basic application in 5 steps
- Other items for migration
- Tips for building & testing application in both modes

# Where to Start

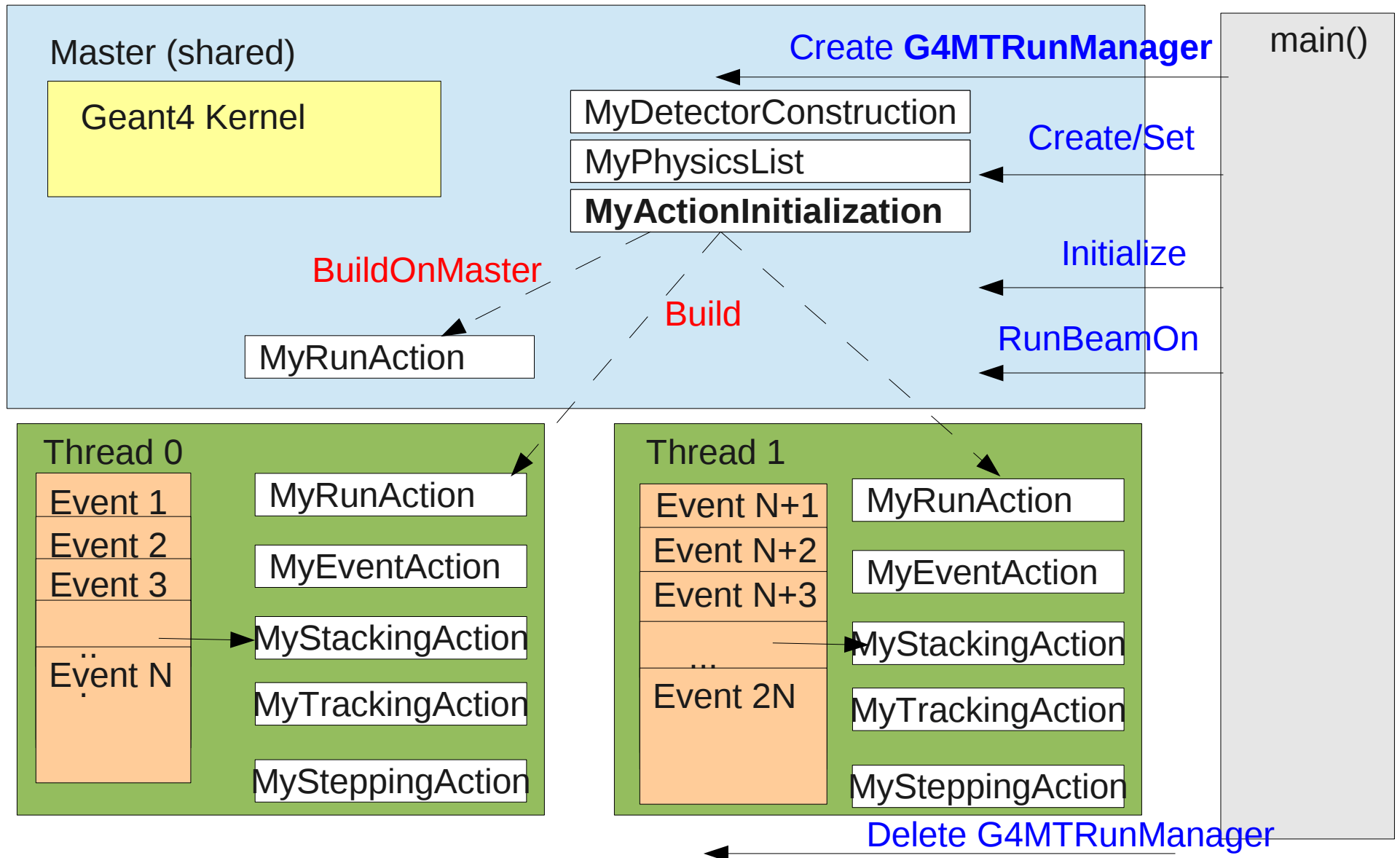
- Basic examples - already migrated to MT
  - Cover most frequently used features
- Geant4 MT For Application Developers wiki page:
  - <https://twiki.cern.ch/twiki/bin/view/Geant4/Geant4MTForApplicationDevelopers>
  - Very detailed instructions covering all aspects
- This presentation

# Understand How It Works

# User Application and Geant4 Kernel In Sequential Mode



# User Application and Geant4 Kernel In MT Mode



# In MT Mode

- It is important to know which objects are thread-local and which are shared
  - For details see [MT guidelines on Wiki page](#)
- When you are in doubt you can quickly check by printing the object pointer:

```
G4cout << "Logical volume: " << fLogicChamber  
      << " SD: " << aTrackerSD << G4endl;
```

- Objects on thread have longer address:

```
G4WT0 > Logical volume: 0xa5feb0 SD: 0x7f5d380e7940
```

# MT Migration of A Basic Application in 5 Steps



# 5 Steps to Migrate Basic Applications

1. Create Action Initialization class
2. Update main()
3. Update Detector Construction class
4. Update G4Allocator declarations (if present)
5. Update analysis code (if present)

# 1. Action Initialization

- Move user actions classes from *main()* to a new *ActionInitialization* class

exampleB1.cc 9.6.p02

```
// Detector construction  
runManager->SetUserInitialization(new B1DetectorConstruction());
```

```
// Physics list  
G4VModularPhysicsList* physicsList = new QBBC;  
runManager->SetUserInitialization(physicsList);
```

Keep in main

```
// Primary generator action  
runManager->SetUserAction(new B1PrimaryGeneratorAction());
```

```
// Stepping action  
runManager->SetUserAction(new B1SteppingAction());
```

```
// Event action  
runManager->SetUserAction(new B1EventAction());
```

```
// Run action  
runManager->SetUserAction(new B1RunAction());
```

Move to new class

# Action Initialization (cont.)

B1ActionInitialization.hh - 10.00

```
#include "G4VUserActionInitialization.hh"

class B1ActionInitialization :
  public G4VUserActionInitialization
{
public:
  B1ActionInitialization();
  virtual ~B1ActionInitialization();

  virtual void BuildForMaster() const;
  virtual void Build() const;
};
```

In MT: both functions are called  
In S: only Build() is called

B1ActionInitialization.cc - 10.00

```
void B1ActionInitialization::BuildForMaster() const
{
  SetUserAction(new B1RunAction);
}

void B1ActionInitialization::Build() const
{
  SetUserAction(new B1PrimaryGeneratorAction);
  SetUserAction(new B1RunAction);
  SetUserAction(new B1EventAction);
  SetUserAction(new B1SteppingAction);
}
```

## 2. main()

- *Instantiate G4MTRunManager and replace the code moved in Action Initialization with use of this class*

exampleB1.cc 10.00

```
// Construct the default run manager
#ifdef G4MULTITHREADED
  G4MTRunManager* runManager = new G4MTRunManager;
  runManager->SetNumberOfThreads(8);
#else
  G4RunManager* runManager = new G4RunManager;
#endif

// Detector construction
runManager->SetUserInitialization(new B1DetectorConstruction());

// Physics list
G4VModularPhysicsList* physicsList = new QBBC;
runManager->SetUserInitialization(physicsList);

// User action initialization
runManager->SetUserInitialization(new B1ActionInitialization());
```

# 3. Detector construction

- Separate creation of **sensitive detectors**, if present, in a new function *CreateSDandField()*

9.6.p02

```
G4VPhysicalVolume* B2bDetectorConstruction::Construct()  
{  
  // ...  
  B2TrackerSD* trackerSD  
    = new B2TrackerSD( "B2/TrackerSD", "TrackerHitsCollection" );  
  G4SDManager::GetSDMpointer()->AddNewDetector( trackerSD );  
  fLogicChamber->SetSensitiveDetector( trackerSD );  
  // ...  
}
```

10.00

```
G4VPhysicalVolume* B2bDetectorConstruction::ConstructSDandField()  
{  
  // ...  
  B2TrackerSD* trackerSD  
    = new B2TrackerSD( "B2/TrackerSD", "TrackerHitsCollection" );  
  G4SDManager::GetSDMpointer()->AddNewDetector( trackerSD );  
  fLogicChamber->SetSensitiveDetector( trackerSD );  
  // ...  
}
```

See basic/B2 example

# Detector construction (cont.)

- A new utility function `G4VUserDetectorConstruction::SetSensitiveDetector` can be also used:

```
G4VPhysicalVolume* B2bDetectorConstruction::ConstructSDandField()
{
  // ...
  B2TrackerSD* trackerSD
    = new B2TrackerSD( "B2/TrackerSD", "TrackerHitsCollection" );
  //G4SDManager::GetSDMpointer()->AddNewDetector( trackerSD );
  //fLogicChamber->SetSensitiveDetector( trackerSD );
  SetSensitiveDetector( "LogicChamberName", trackerSD );
  // or
  SetSensitiveDetector( fLogicChamber, trackerSD );
  // ...
}
```

- Be careful if you define messenger classes to instantiate them at the right function
  - Messengers handling geometry data can stay in the constructor
  - Messenger handling SD (or Field) data should be also instantiated in the new method

# Detector construction (cont.)

- Separate creation of a **magnetic field**, if present, in a new function `CreateSDandField()`

9.6.p02

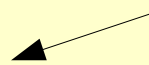
```
B2bDetectorConstruction::B2bDetectorConstruction() {  
    // ...  
    fMessenger = new B2bDetectorMessenger(this);  
    fMagField = new B2MagneticField();  
}
```

ref-07

```
G4VPhysicalVolume* B2bDetectorConstruction::ConstructSDandField() {  
    // ...  
    fMagField = new B2MagneticField();  
    // ...  
}
```

```
B2MagneticField::B2MagneticField() {  
    // ...  
    fMagField = new B2FieldMessenger();  
    // ...  
}
```

New magnetic field messenger class separated from B2bDetectorMessenger



# Detector construction (cont.)

- A new global field messenger class is available to define a uniform magnetic field with an interactive command for setting the field value
  - No B2 specific Magnetic field and Magnetic Field Messenger classes are needed anymore

10.00

```
G4VPhysicalVolume* B2bDetectorConstruction::ConstructSDandField() {  
  // ...  
  G4ThreeVector fieldValue = G4ThreeVector();  
  fMagFieldMessenger = new G4GlobalMagFieldMessenger(fieldValue);  
}
```

See basic/B2 example

- If the field class has a cache mechanism, such field class object must be thread-local
  - See MT guidelines on Wiki page for an example



# 4. G4Allocator

- Change *static* declarations to *static G4ThreadLocal*
- Typically G4Allocator is used in hit, trajectory and trajectory point classes

B2TrackerHit.hh 9.6.p02

```
extern G4Allocator<B2TrackerHit>* B2TrackerHitAllocator;
```

B2TrackerHit.cc 9.6.p02

```
G4Allocator<B2TrackerHit>* B2TrackerHitAllocator=0;
```

B2TrackerHit.hh 10.00

```
extern G4ThreadLocal G4Allocator<B2TrackerHit> B2TrackerHitAllocator;
```

B2TrackerHit.cc 10.00

```
G4ThreadLocal G4Allocator<B2TrackerHit>* B2TrackerHitAllocator=0;
```

# 5. Analysis

- *Analysis manager has to be created using a new method `G4AnalysisManager::Create(G4bool isOnMaster)`*

9.6.p02

```
void B4RunAction::BeginOfRunAction(const G4Run* run) {  
    ...  
    // Create analysis manager  
    G4AnalysisManager* manager = G4AnalysisManager::Instance();  
    ...  
}
```

10.00

```
B4RunAction::B4RunAction(G4bool isOnMaster) {  
    ...  
    // Create analysis manager  
    G4AnalysisManager* manager = G4AnalysisManager::Create(isOnMaster);  
    ...  
}
```

# MT Migration of More Advanced Applications

# Other Items For Migration (1)

- Scoring in user action classes
  - Derived class of G4Run should be used to collect data over threads at the end of an event loop
  - See [basic examples B1, B3 and B4/B4b](#)
- User defined physics list
  - G4GenericIon::GenericIonDefinition() has to be added in ConstructParticle()
  - All process objects have to be instantiated in ConstructProcess(), which is invoked for every thread
  - See [extended/electromagnetic/TestEm4](#)
- Random numbers
  - CLHEP::Random:: must be replaced with [G4Random::](#)

# Other Items For Migration (2)

- Primary generation action reading input data file
  - Mutex locking mechanism is needed to share the same input data file on threads
  - See [extended/runAndEvent/RE05](#)
- Static declarations in user code
  - Be sure that all static variables are used from threads as read-only
- Customize threading
  - Via `G4UserWorkerInitialization` class
- Advanced Tips & Tricks
  - <https://twiki.cern.ch/twiki/bin/view/Geant4/Geant4MTTipsAndTricks>

# Tips For Building & Testing Application in Both Modes

# Building Geant4

- Install Geant4 in both modes:
  - Keep a single version of geant4 source and create two build directories:
    - geant4\_version\_build, geant4\_version\_build\_mt
  - Build geant4 in each directory **with the same cmake options** but only added **-DGEANT4\_BUILD\_MULTITHREADED=ON** for MT build
    - In some cases more options may be needed for MT build
  - Install each version in its installation area
    - geant4\_version\_install, geant4\_version\_install\_mt

# Building Application

- Install your application (example) in both modes
  - Create two build directories:
    - my\_example\_build, my\_example\_build\_mt
  - Build the application (example) from the same source against the relevant Geant4 installation:

```
cd my_example_build  
cmake -DGeant4_DIR=/mypath/geant4_version_install/lib64/Geant4-10.0  
/mypath/geant4_source/examples/my_example
```

```
cd my_example_build_mt  
cmake -DGeant4_DIR=/mypath/geant4_version_install_mt/lib64/Geant4-10.0  
/mypath/geant4_source/examples/my_example
```

- You can also build each examples category or sub-category at once



# Testing Application

- Run your application in both build directories and compare the produced output (output analysis files, logs)
  - Tips: you can redirect output from each thread in a file (and avoid their inter-laying):
    - `/control/cout/setCoutFile fileName [ifAppend]`
    - `/control/cerr/setCerrFile fileName [ifAppend]`
  - It is worth to test your application with different number of threads
  - gdb is working fine with MT mode, useful in case of crashes