Enhancements to multiprocessing in ROOT

Enrico Guiraud

supervisor:
Gerardo Ganis
New lightweight framework for multi-process applications

ROOT session

ROOT session (worker)

ROOT session (worker)

ROOT session (worker)

ROOT session (worker)
The *old* way to create workers

```c
// The old way to create workers

gSystem->Exec("/bin/prooefserv &")
```
...the new way to create workers
Advantages of the new approach

➔ versatility: workers have access to the session’s environment

➔ small footprint: workers’ memory is copy-on-write

➔ starting and stopping workers is cheap and fast

Disadvantage

➔ portability

https://github/bluehood/root
A new type of workflow is possible!

- fork()
- fork and connect to servers
- objects are sent back and forth using ROOT's streaming capabilities

- normal workflow
- reap subprocesses
- collect results
- distribute tasks

[https://github/bluehood/root](https://github/bluehood/root)
An application: the Map function

```
TPool pool

auto result = pool.Map( func, arguments )
```

```
| func(2) | 1 |
| func(3) | 2 |
| func(1) | 3 |
```
An application: the Map function

```
TPool pool(8)

auto result = pool.Map( func, arguments )
```

defaults to n. of cores
An application: the Map function

```cpp
TPool pool(8)

auto result = pool.Map(func, arguments)
```

C/C++ function loaded macro functor class std::function lambda expr.

defaults to n. of cores

https://github/bluehood/root
An application: the Map function

```cpp
TPool pool(8)

auto result = pool.Map( func, arguments )
```

- C/C++ function
- loaded macro
- functor class
- std::function
- lambda expr.

- std::container
- initializer list
- TCollection*
An application: the Map function

```cpp
TPool pool(8)

auto result = pool.Map(func, arguments)
```

- `std::vector`
- `TObjArray`
- `std::container` (initializer list)
- `TCollection*`
- C/C++ function
- loaded macro
- functor class
- `std::function`
- lambda expr.

defaults to n. of cores
An application: the Map function

What about Reduce?

1
2
3

Map

func(2)
func(3)
func(1)

Reduce

func(2) + func(3) + func(1)

https://github/bluehood/root
An application: the Map function

What about Reduce?

MapReduce(func, args, reduce_func)
Thank you for your time!
Thank you for having me *here*!
Implementation details

Client
- Fork
- Broadcast/Send
- Collect

Server
- virtual Handle Input

TFileHandler

Note<T>
- int code
- T object

Generic Code
- message
- error
- fatal error
- shutdown order
- shutdown notice

https://github/bluehood/root
Implementation details: Map

Client
- Fork
- Broadcast/Send
- Collect
- virtual Handle Input

Server
- virtual Handle Input

Pool
- Map
- Handle Input

Note<T>
- int code
- T object

Pool Code
- execute func
- func result

Pool Server<F, T>
- Handle Input
- F function
- T arguments

TFileHandler
An application: the Map function

```cpp
TPool pool(8)

.L myMacro.cxx+

auto res = pool.Map(  
    [] (string f) { return myMacro("opt", 12, f); },  
    {"file1", "file2", "file3"}  
)
```

https://github/bluehood/root
Map speed-up

This is the speed-up on filling an histogram with 1E9 random numbers

N.B. reduce overhead is not shown
MapReduce speed-up