OC4MC

OCaml for MultiCore

Deuxième réunion du GDR GPL/LTP

LACL / Université Paris XII

21 Octobre 2009

Mathias Bourgoin, Adrien Jonquet
Benjamin Canou, Philippe Wang
Emmanuel Chailloux

project timeline

2008

June

September

2009

April

Now

Jane Street OCaml summer project

(pause)

Mathias' research project

continuing

broken implementations

numerous bug fixes

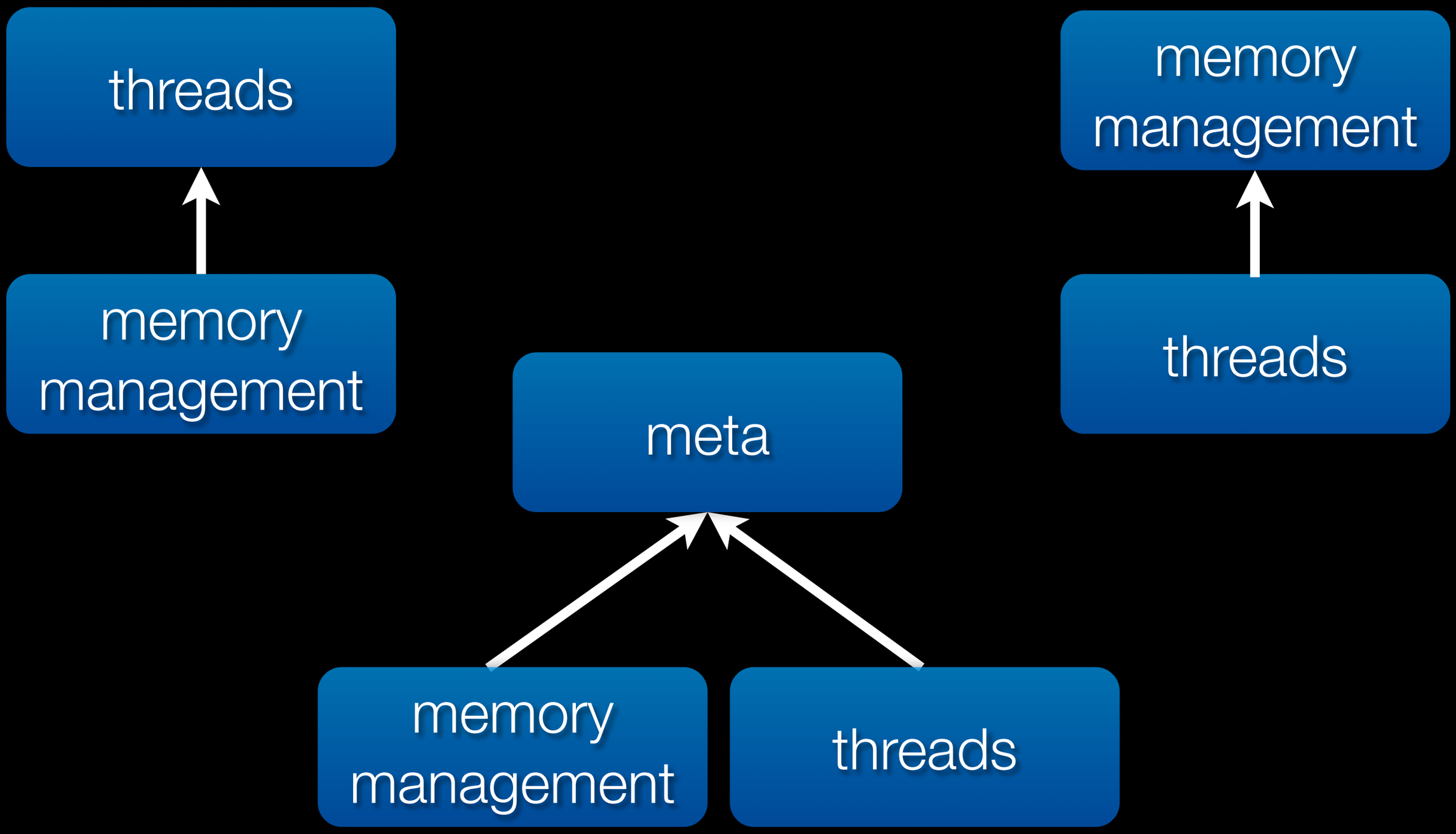
multithreading based multicore profit with OCaml

- (super light) preliminaries
- chapter 1 / OCaml & Parallelism
- chapter 2 / OCaml for MultiCore
- chapter 3 / Performance
- chapter 4 / Conclusion & Future Work

preliminaries

dependencies mess

dependencies



interdependencies

if there is a garbage collector and there are threads, then their implementations are probably interdependent

chapter 1

OCaml & Parallelism

(what a mess around)

multitask programming

join calculus
concurrent dataflow
cooperative threads
preemptive threads
ASM
cluster
CUDA
grid
Ada
(heavy) process
lo
OpenCL
C
p2p
Oz
SML
OCaml
Java
Concurrent ML
internet computing
JOCaml
Alice
Haskell
Join Java
Concurrent Haskell
MultiLisp
Scala
Clojure
PPE+SPEs
assembly tweaking
shared memory
message passing
CELL
CCS
SIMD GPU
MIMD CPU
π-calculus
NVIDIA GeForce GT130
Intel Core Quad
pict

(still the same mess)

multitask programming

Concurrent Haskell
JOCaml
Concurrent ML
Join Java
π-calculus
Alice
pict
CCS
Io
MultiLisp
Ada
Oz
OC4MC
OCaml
Haskell
SML
Scala
join calculus
concurrent dataflow
Clojure
Java
OpenCL
CUDA
cooperative threads
internet computing
preemptive threads
p2p
C
(heavy) process
cluster
message passing
ASM
assembly tweaking
shared memory
grid
SIMD GPU
PPE+SPEs
MIMD CPU
NVIDIA GeForce GT130
CELL
Intel Core Quad

some Caml history

cheap multicores

1985 1991 1996 today

Caml

Caml Light

Objective Caml

Objective Caml

- Functional-based multiparadigm language
- Distribution by INRIA
  - known for its efficiency
  - non-parallel concurrent threads

some

parallel threads issues

- runtime library support
- reentrance (beware of shared static variables)
- memory allocation/collection

addressing some

parallel threads issues

- runtime library support
  - reentrance (beware of shared static variables)
    - use (POSIX) `__thread` facility
    - transform to function parameters
    - refactoring
- memory allocation/collection

addressing some

parallel threads issues

- runtime library support
  - reentrance (beware of shared static variables)
- memory allocation/collection
  - look at the guts, be scared, run or make a choice
    - learn, understand, adapt
    - learn, remove parts & rewrite from scratch

INRIA OCAML

memory management

- **allocation**
  - `heap_ptr = heap_ptr - size(val)`
- **two-generation collection**
  - minor collection: stop&copy
  - major collection: incremental mark&sweep&compact

INRIA OCAML

garbage collection

young heap

Stop&Copy
survivors
to old heap

old heap

Mark&Sweep
+ compact

very efficient

generational

incremental

(runtime lib implementation in C code + asm)

chapter 2

OCaml for MultiCore

(influence of the) 


past experience

Pagano et al. (JFLA 2007, PADL 2008, ICFP 2009)

- alternative runtime lib implementation for software certification required by civil avionics norms
- memory management too hard to explain
  thus impossible to certify
- remove concurrency, marshalling, weak pointers, ...
- result: from 16 000 lines of C code to 4 500

addressing some

parallel threads issues

- runtime library support
  - reentrance (beware of shared static variables)
  - memory allocation/collection
    - look at the guts, be scared, run or make a choice
      - learn, understand, adapt
      - learn, remove parts & rewrite from scratch

and what about scheduling?

interdependencies

if the memory management implies moving values,
then it can stop a thread from accessing values...

OCaml for MultiCore

- **new memory management**
  - **stop** the world **&** two-generation **copy**
    - “inherited” mechanisms
      - stop at allocation, blocking operations
- **parallel threads**

OC4MC

Partial Collection

Stop&Copy

pages

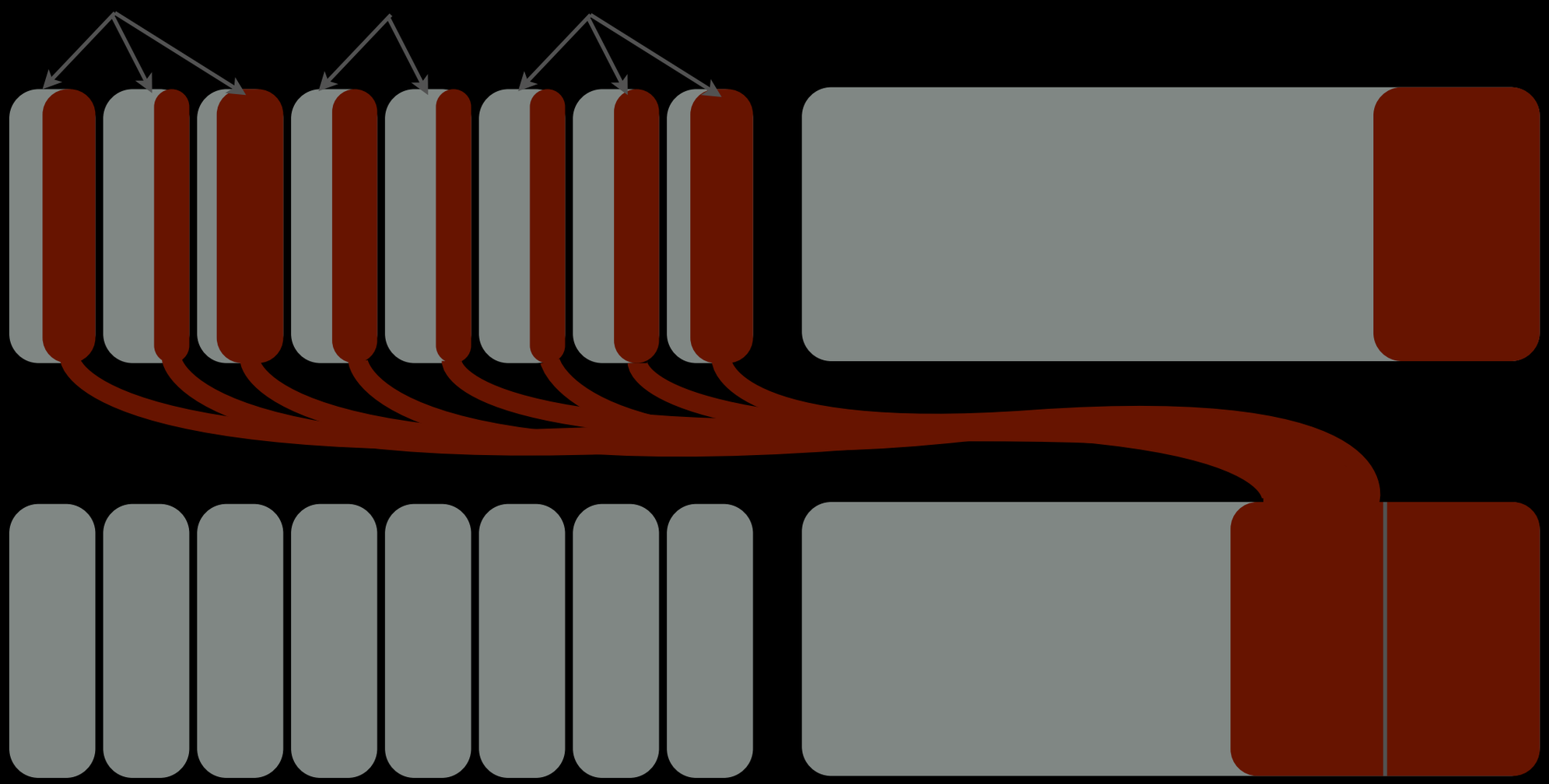
shared heap

thread 1 thread 2 thread 3

Before

Partial GC

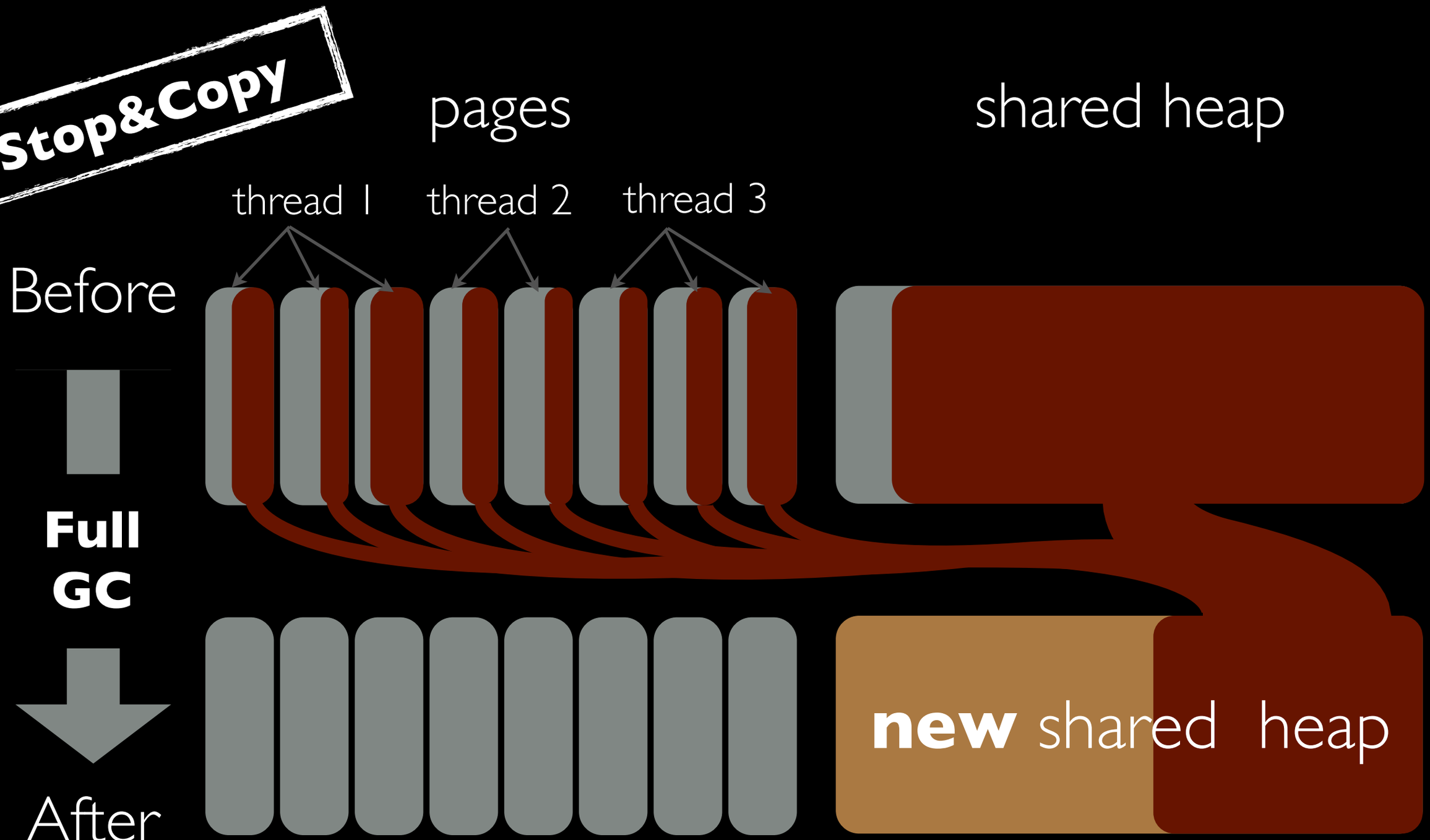
After



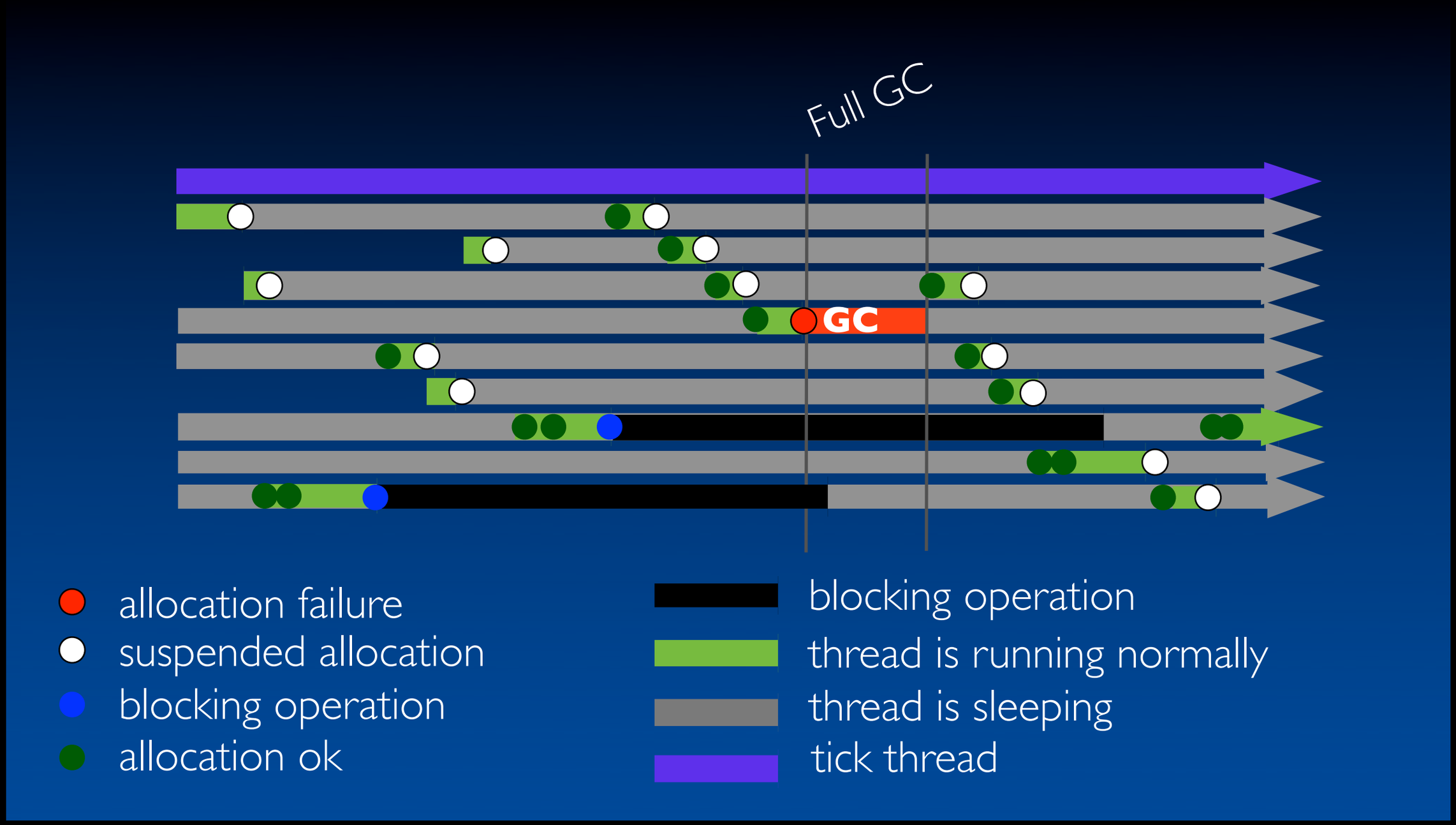
OC4MC

Full Collection

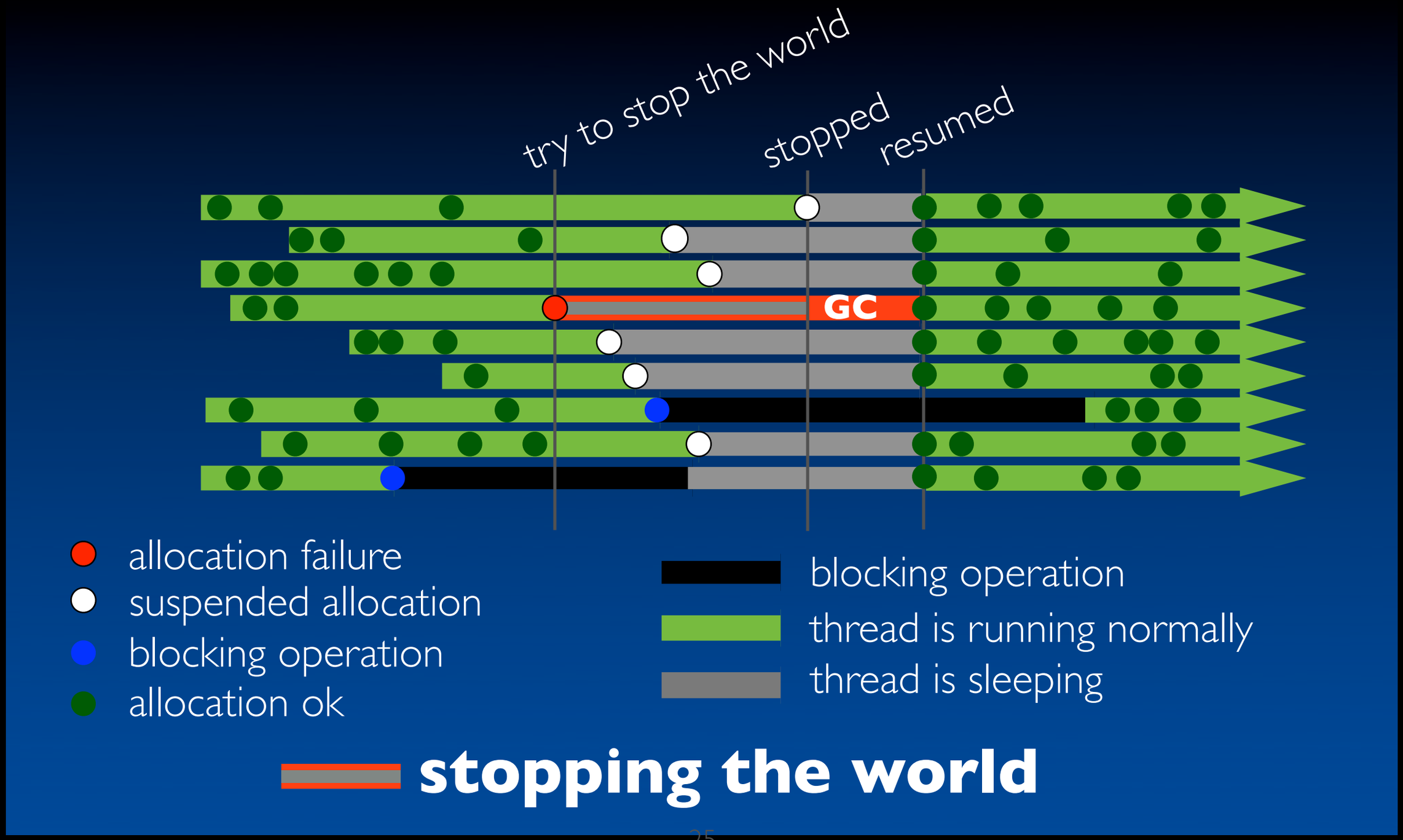
Stop&Copy



Execution Sample



Execution Sample



chapter 3

performance

Speedups

	O'Caml	**OC4MC**				
# of threads	1TH	1TH	2TH	4TH	8TH	16TH
Sieve	60s	64s	32s	16s	10s	9s
speedup	**1.06**	**1**	**2**	**4**	**6.4**	**7.11**
Matmult	15.5s	18.2s	9.5s	4.7s	2.5s	2.5s
speedup	**1.17**	**1**	**1.92**	**3.88**	**7.28**	**7.28**
Life	24.3s	24.7s	16.6s	13.7s	15.1s	15.2s
speedup	**1.02**	**1**	**1.49**	**1.80**	**1.64**	**1.62**

little number of threads

	O'Caml	**OC4MC**
Sieve CML-style	89s	59s
speedup	**1**	**1.50**

great number of threads

X86-64 Quad Core X 2
Virtual Machine : 7 active cores

Slowdowns

- GC algorithm kept as simple as possible
  - heap growth can slow down the program
    - very functional style (many short life objects) combined with some long life objects shows stop&copy weakness
- Predictable weakness

chapter 4

conclusion & future work

Conclusion

it’s working!

http://www.algo-prog.info/ocmc/

open source distribution

- Linux x86 64-bit
- Alternative multicore capable runtime library
  - memory manager replacement
  - thread library replacement
- Available as a patch for OCaml 3.10.2
  - **http://www.algo-prog.info/ocmc/**

Conclusion

- working multicore capable threads for OCaml
- proof of feasibility
- potential good performance

http://www.algo-prog.info/ocmc/

Related & Future Work

- OCaml concurrent/parallel extensions
  - what if they are used with OC4MC
- Use a limited number of threads by implementing abstractions on system threads
  - e.g. Parallel Concurrent ML
- Alternative GC algorithm

http://www.algo-prog.info/ocmc/