Time-Portable Programming the JAviator in the Tiptoe VM

Christoph Kirsch
Universität Salzburg

UC Berkeley
January 2009

The JAviator

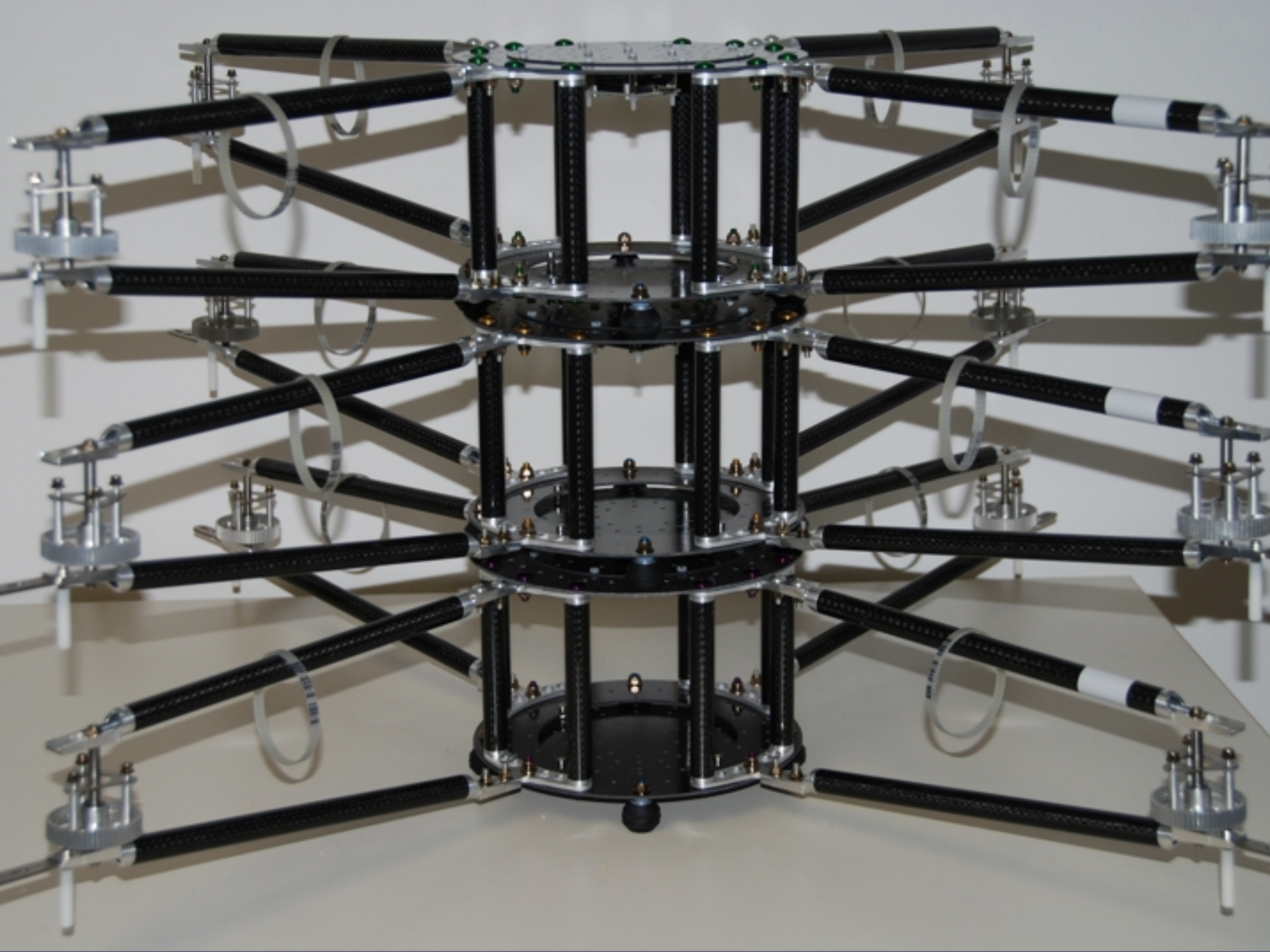
javiator.cs.uni-salzburg.at

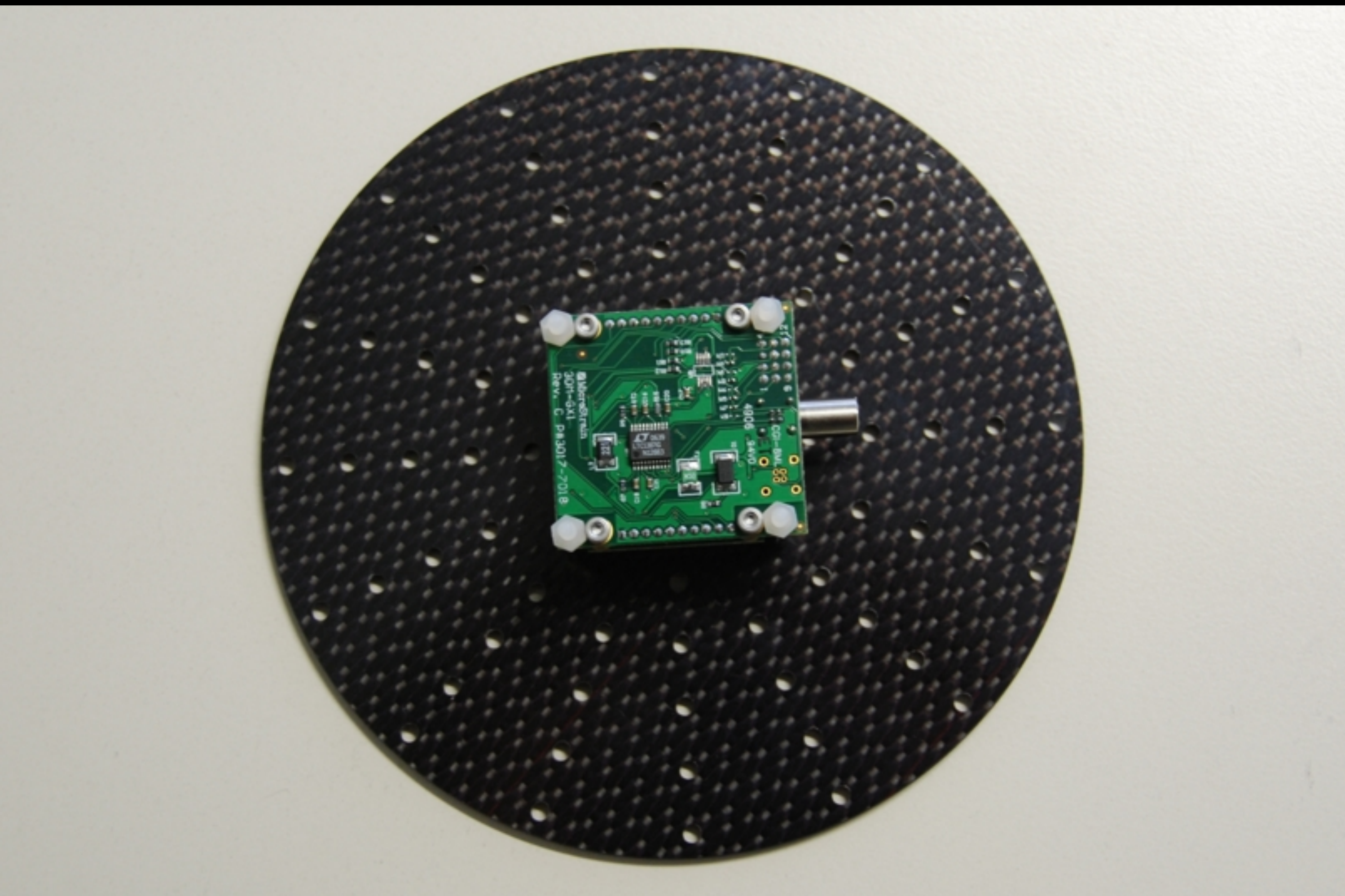
javiator.cs.uni-salzburg.at#

- Silviu Craciunas* (Control Systems)
- Harald Röck (Operating Systems)
- Rainer Trummer (Frame, Electronics)

#Supported by a 2007 IBM Faculty Award and the EU ArtistDesign Network of Excellence on Embedded Systems Design
*Supported by Austrian Science Fund Project P18913-N15

Quad-Rotor Helicopter

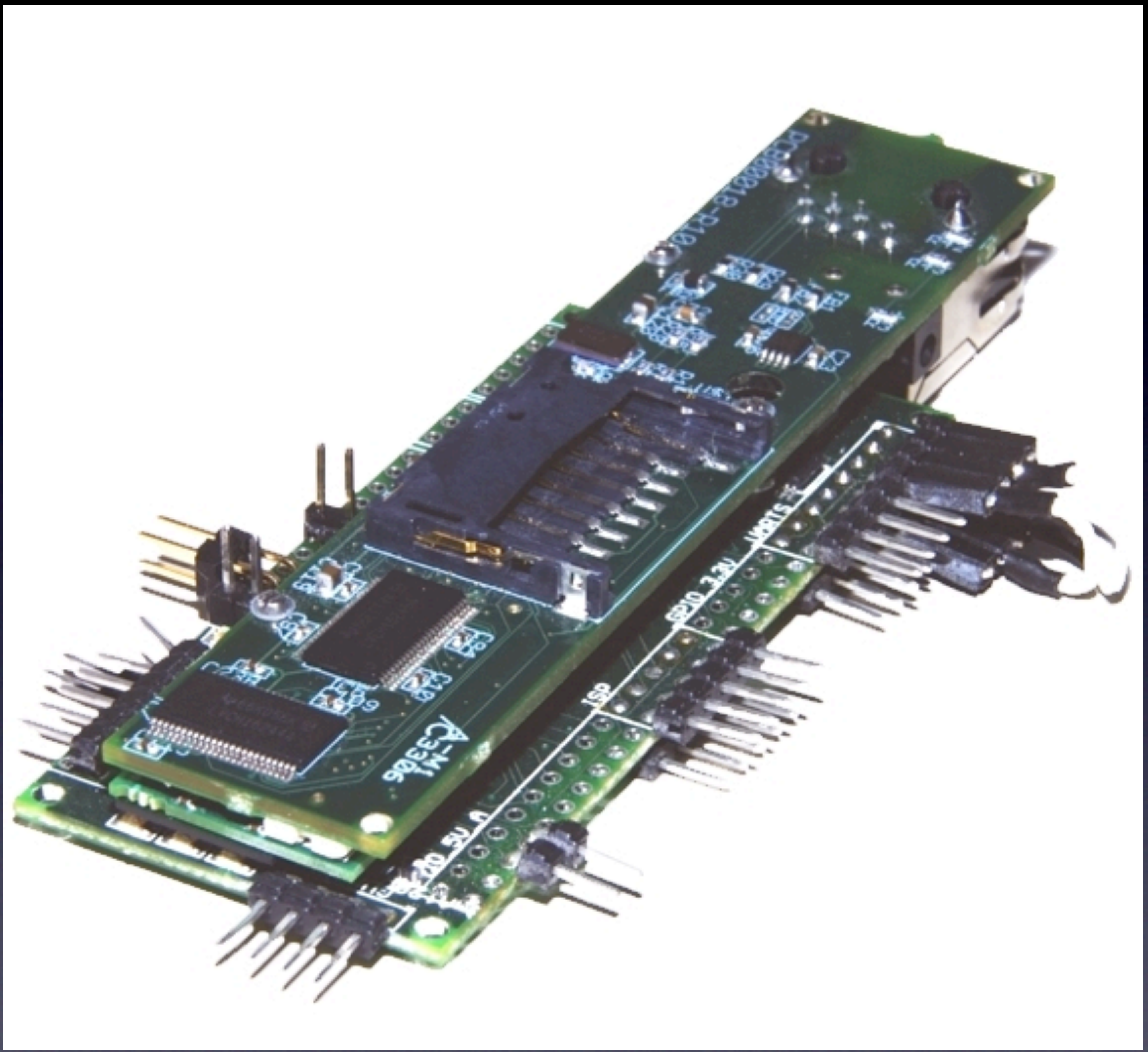




Gyro

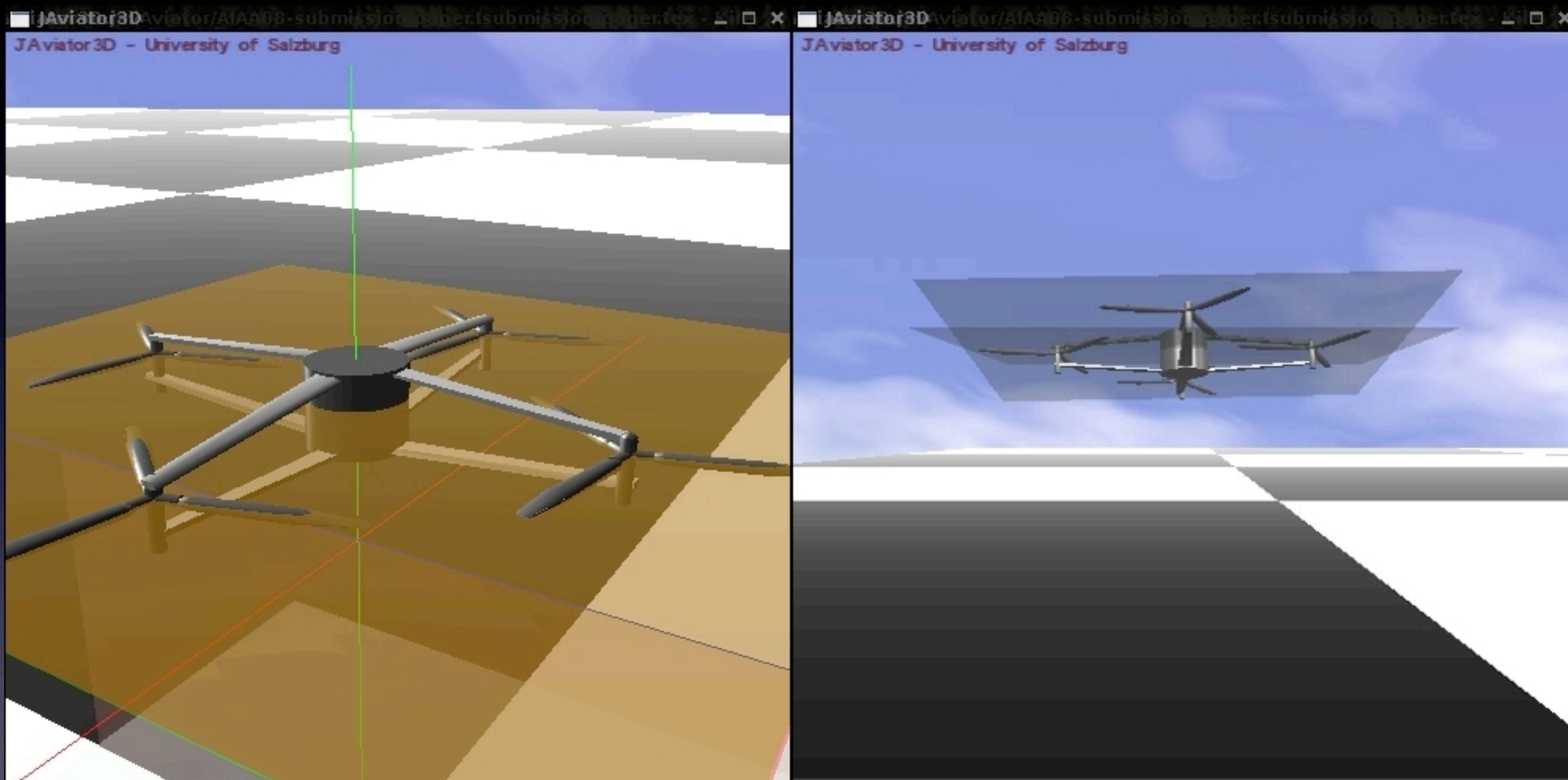
Propulsion

Gumstix

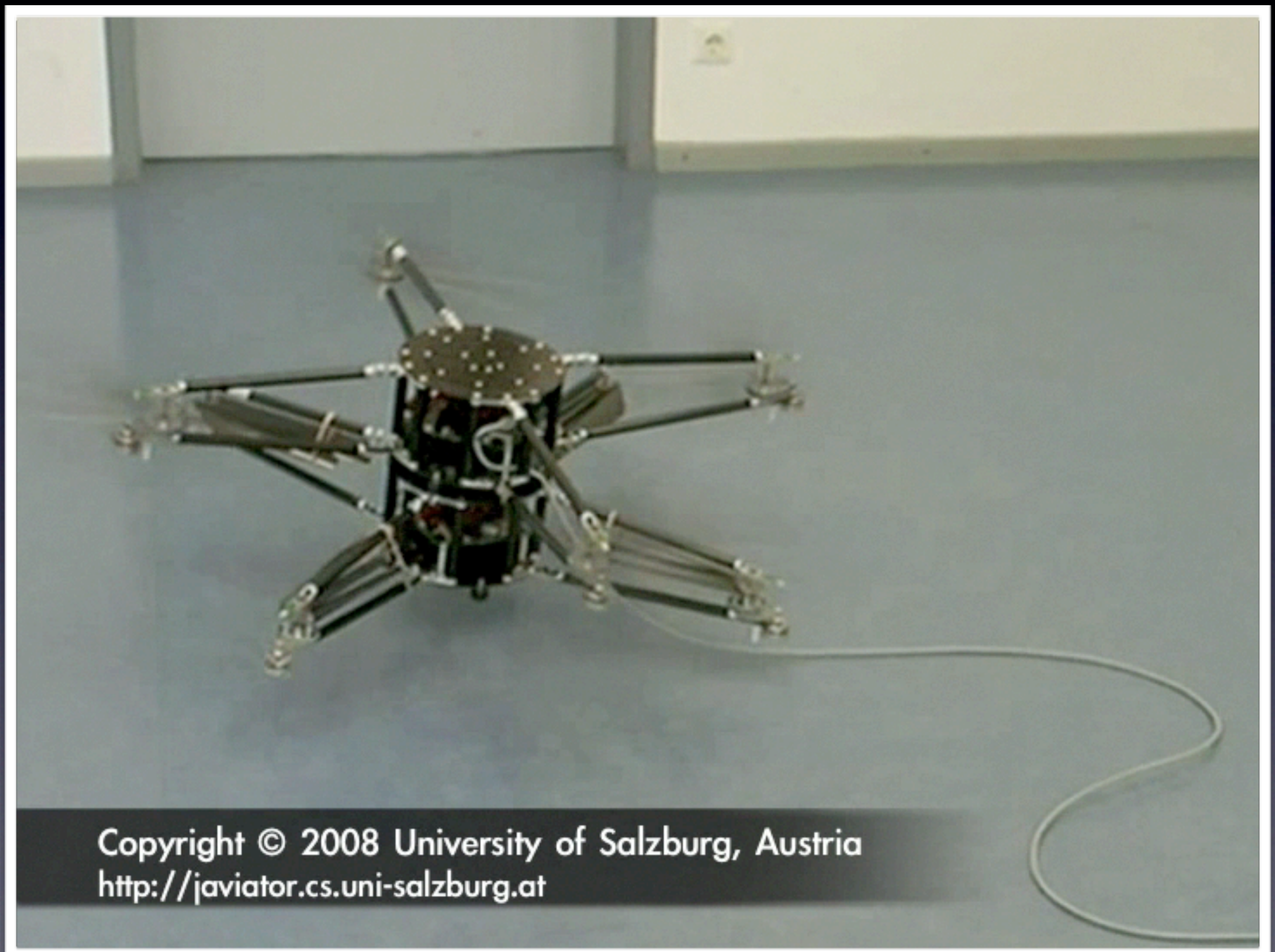


600MHz XScale, 128MB RAM, WLAN, Atmega uController

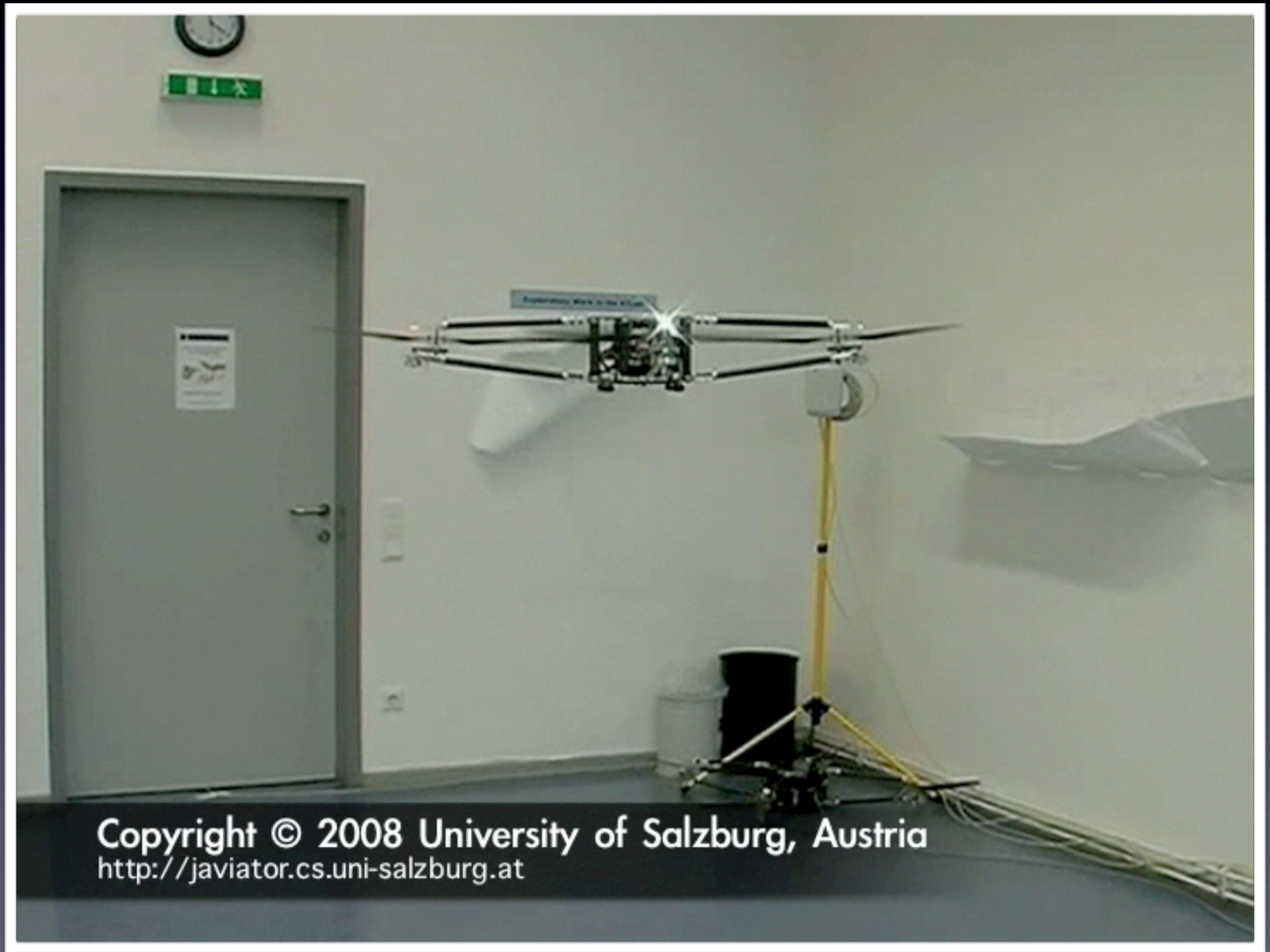

JAviator Control Terminal
Yaw Limit
Joystick Mode
Direct Connection
Connect to Heli
Start Test Mode
Start Logging
Roll Limit
Yaw
deg
Roll
deg
Front
Left
Hover
Right
Rear
-1
-1
0
1
Offsets
1
33 Yaw Trim
Roll Trim 0
0
-1
0
Pitch
deg
Altitude
cm
351
353
Signals
352
351
Pitch Trim 0
Altitude Limit
Port Settings
Reset Needles
Shut down Heli
Key Assistance
About Terminal
Pitch Limit


JAviator3D
JAviator3D - University of Salzburg
JAviator3D
JAviator3D - University of Salzburg

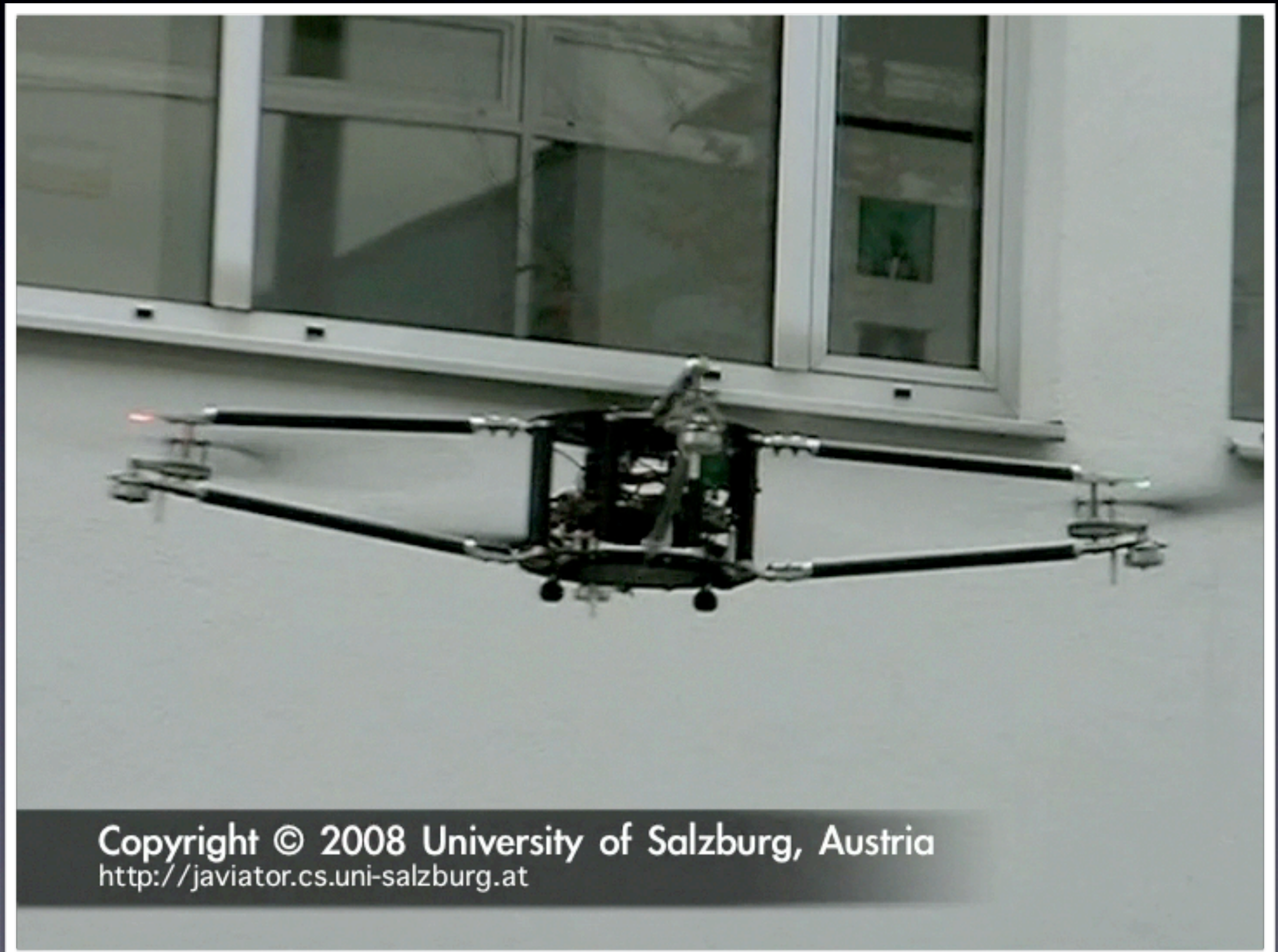
[AIAA GNC 2008]



Indoor Flight
STARMAC Controller



Outdoor Flight STARMAC Controller

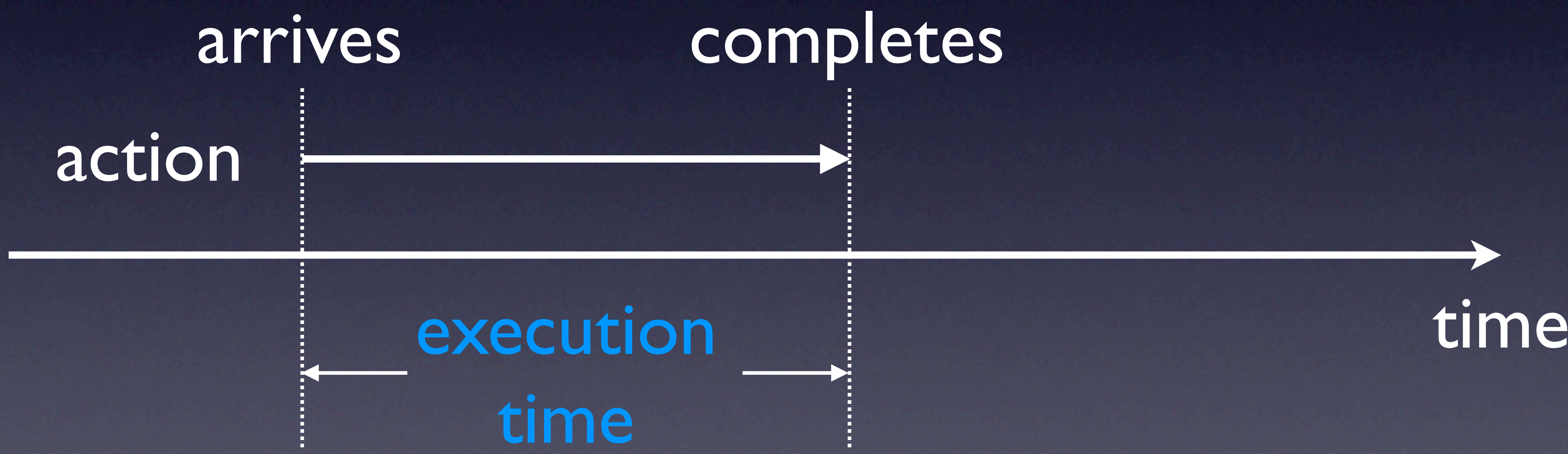


Outdoor Flight
Salzburg Controller

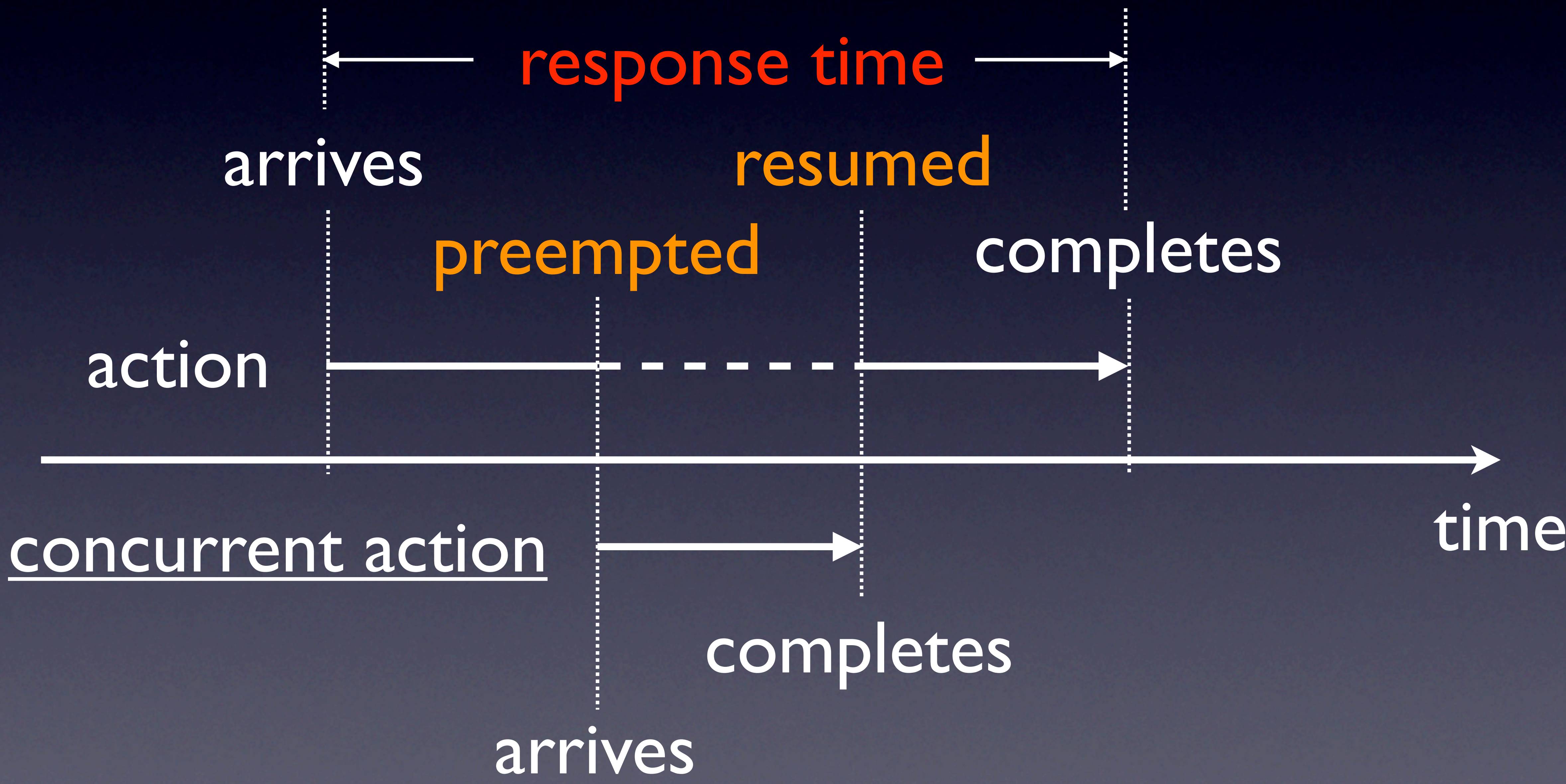
Outline

1. Time-Portable Programming
2. Tiptoe VM Scheduler
3. Tiptoe VM Memory Management

Process Action



Concurrency

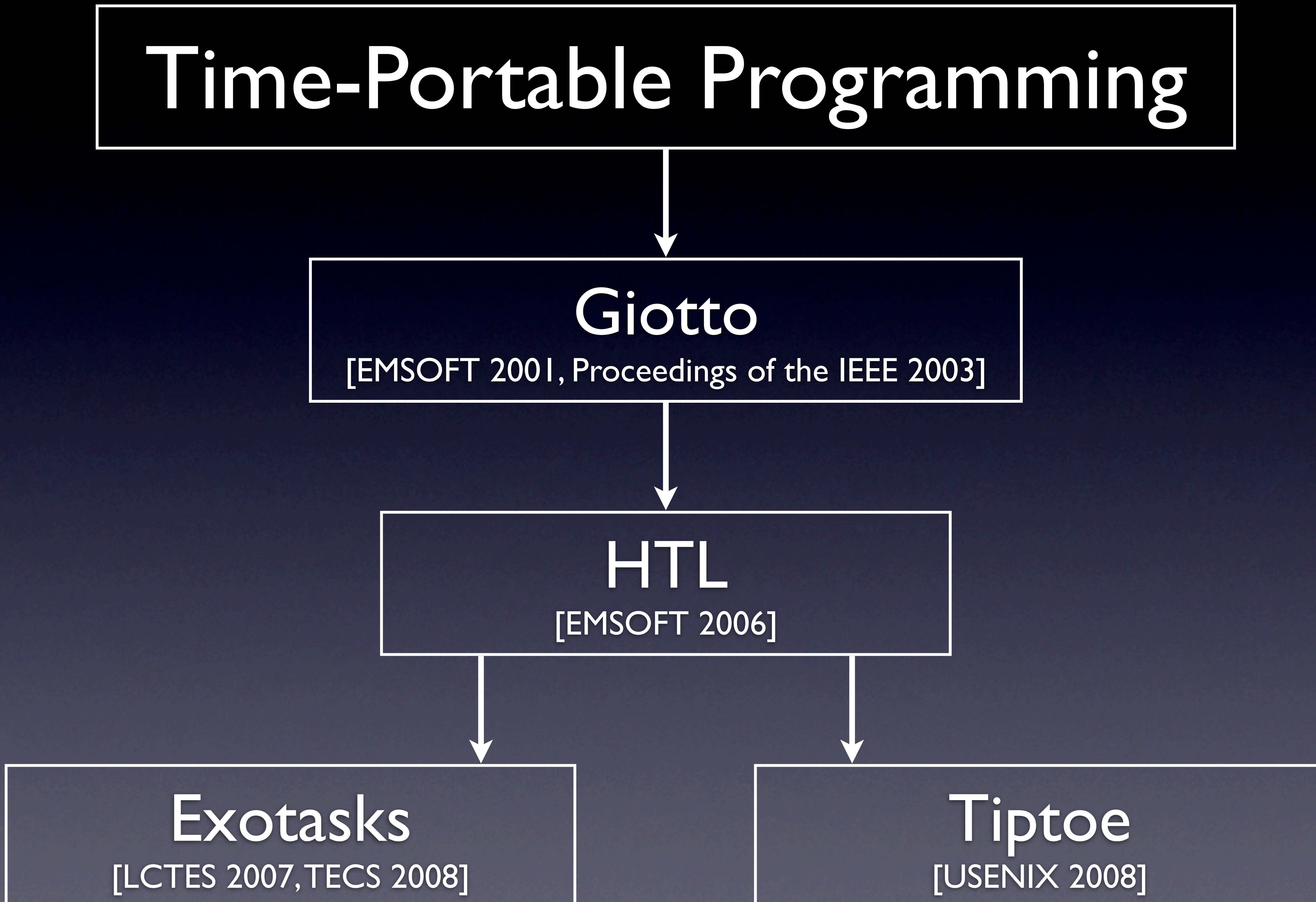


Time

- The temporal behavior of a process action is characterized by its execution time and its response time
- The execution time is the time it takes to execute the action in the absence of concurrent activities
- The response time is the time it takes to execute the action in the presence of concurrent activities

Time-Portable Programming

- Time-portable programming specifies and implements upper AND lower bounds on response times of process actions
- A program is time-portable if the response times of its process actions are maintained across different hardware platforms and software workloads
- The difference ε between upper and lower bounds is its "degree of time portability"


Time-Portable Programming
Giotto
[EMSOFT 2001, Proceedings of the IEEE 2003]
HTL
[EMSOFT 2006]
Exotasks
[LCTES 2007, TECS 2008]
Tiptoe
[USENIX 2008]

Tiptoe: Bare-Metal VM

- OS vs. VM = Processes vs. Processors
- Tiptoe is a VM, will be a kernel-based hypervisor
- Tiptoe virtualizes embedded processors, byte code interpreters in real time
- Tiptoe controls throughput and latency of CPU, memory, and I/O
- I/O is multiplexed onto a collision-free, point-to-point Ethernet link to an I/O host

Outline

1. Time-Portable Programming
2. Tiptoe VM Scheduler
3. Tiptoe VM Memory Management

tiptoe.cs.uni-salzburg.at#

- Silviu Craciunas* (Programming Model)
- Hannes Payer* (Memory Management)
- Harald Röck (VM, Scheduling)
- Ana Sokolova* (Theoretical Foundation)
- Horst Stadler (I/O Subsystem)

#Supported by a 2007 IBM Faculty Award and the EU ArtistDesign Network of Excellence on Embedded Systems Design
*Supported by Austrian Science Fund Project P18913-N15

Variable-Bandwidth Servers

[submitted]

- Tiptoe uses variable-bandwidth servers (VBS)
- VBS generalize constant-bandwidth servers (CBS)
- A CBS executes a process for λ units of time every π units of time
- The pair (λ,π) is called a virtual periodic resource
- A VBS merely has a utilization bound (bandwidth cap) in percentage of CPU time

Result: Programmable Temporal Isolation

- A process executing on a VBS can switch (from one process action to the next) to any virtual periodic resource with a CPU utilization λ/π less or equal to the VBS' utilization bound
- Theorem:
  - The response times of any given process action of any given process can vary at most by π, if the sum of the utilization bounds of all VBS in the system is less or equal to 100%

The smaller the π
the smaller the ε may be,
that is, the higher the
"degree of time portability"
but also
the higher the
scheduling overhead

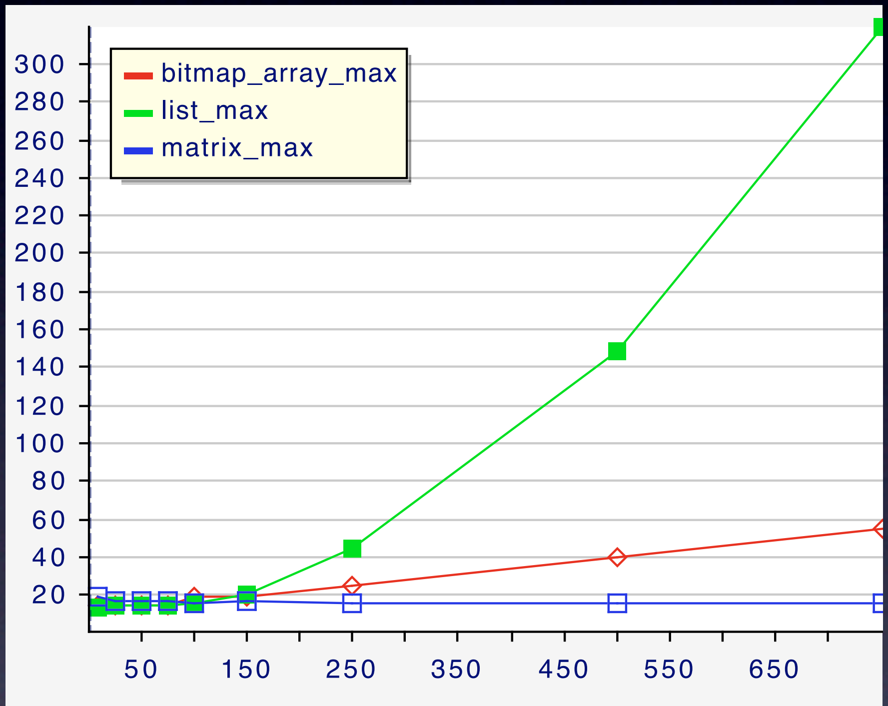
Admission and Scheduling

- Process admission:
  - How do we efficiently test schedulability of newly arriving processes
- Process scheduling:
  - How do we efficiently schedule processes on the level of individual process actions?

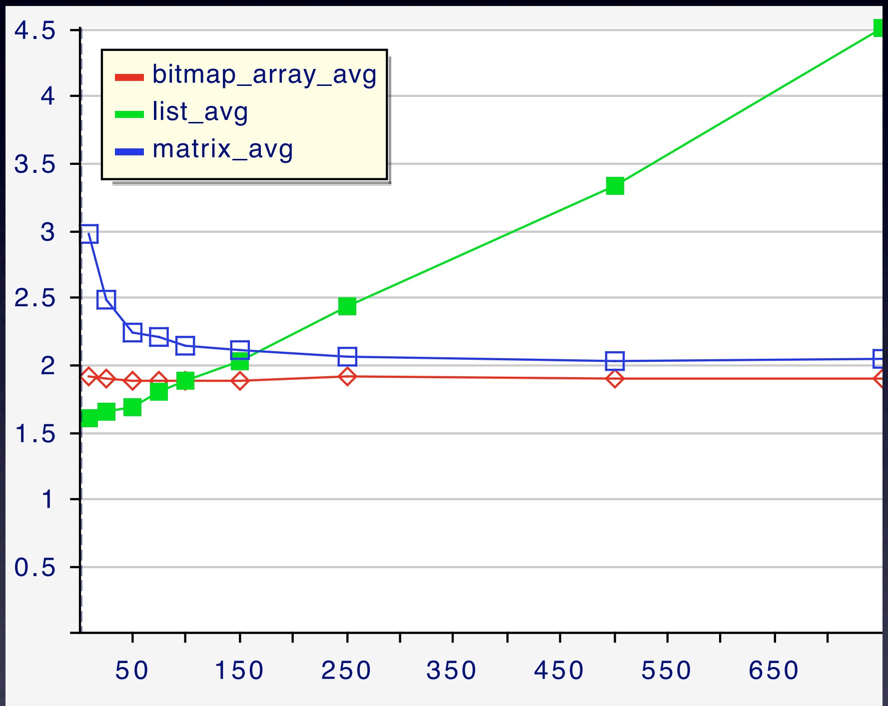
Scheduling Algorithm

- maintains a queue of ready processes ordered by deadline and a queue of blocked processes ordered by release times
- ordered-insert processes into queues
- select-first processes in queues
- release processes by moving and sorting them from one queue to another queue

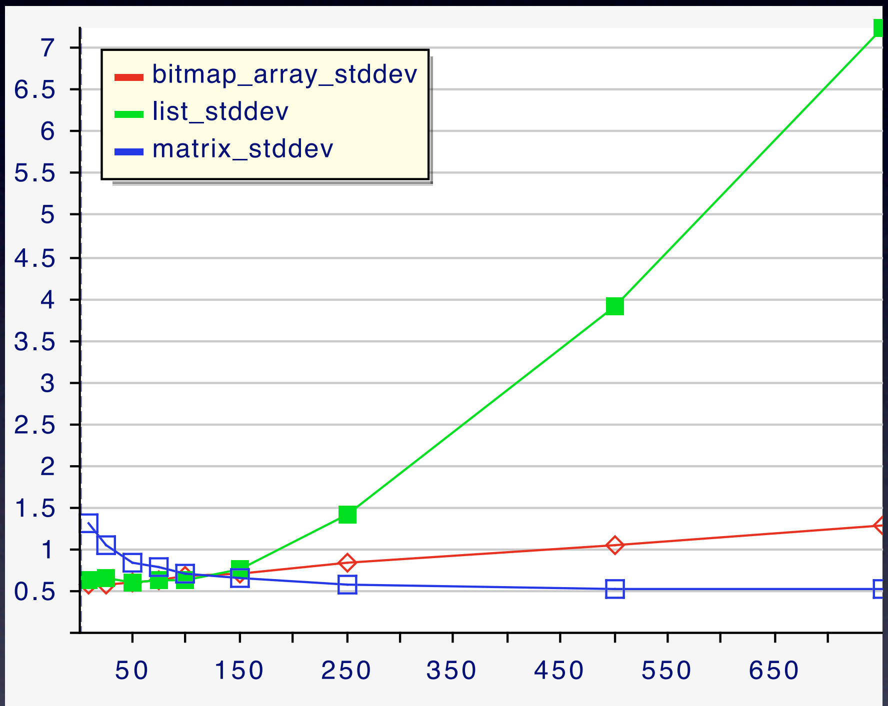
Scheduler Overhead



Max

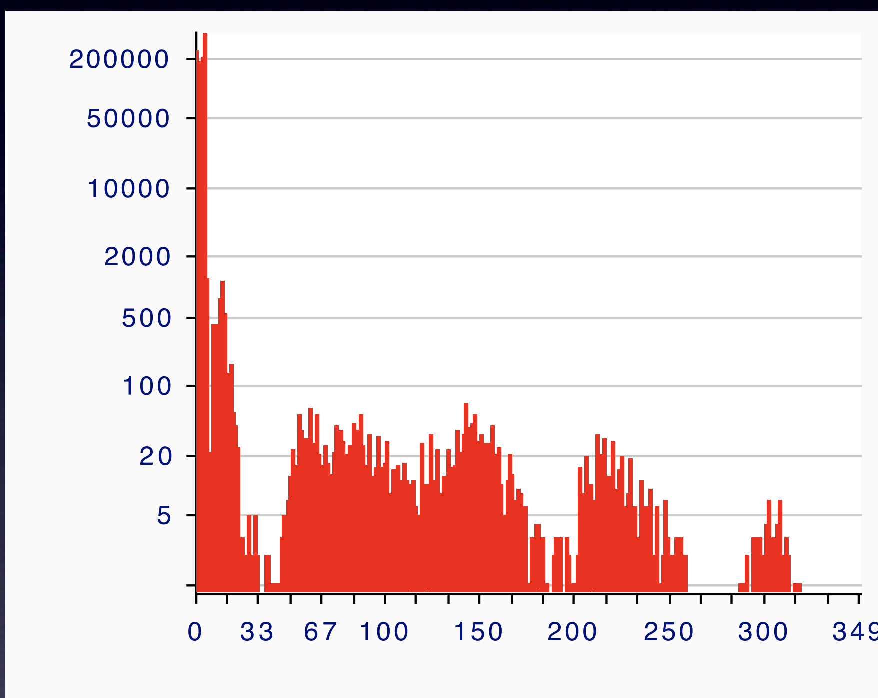


Average

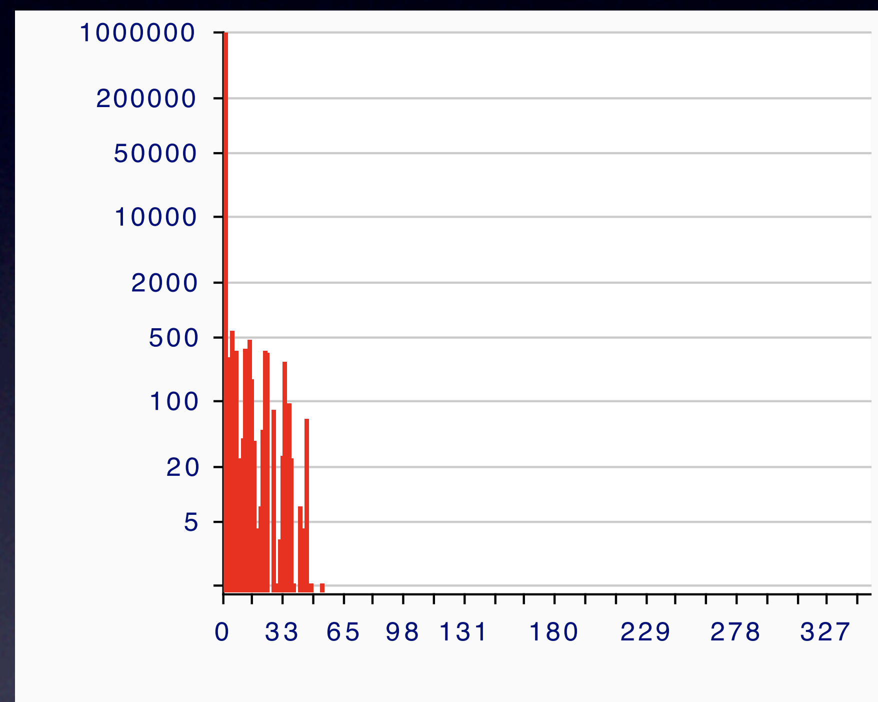


Jitter

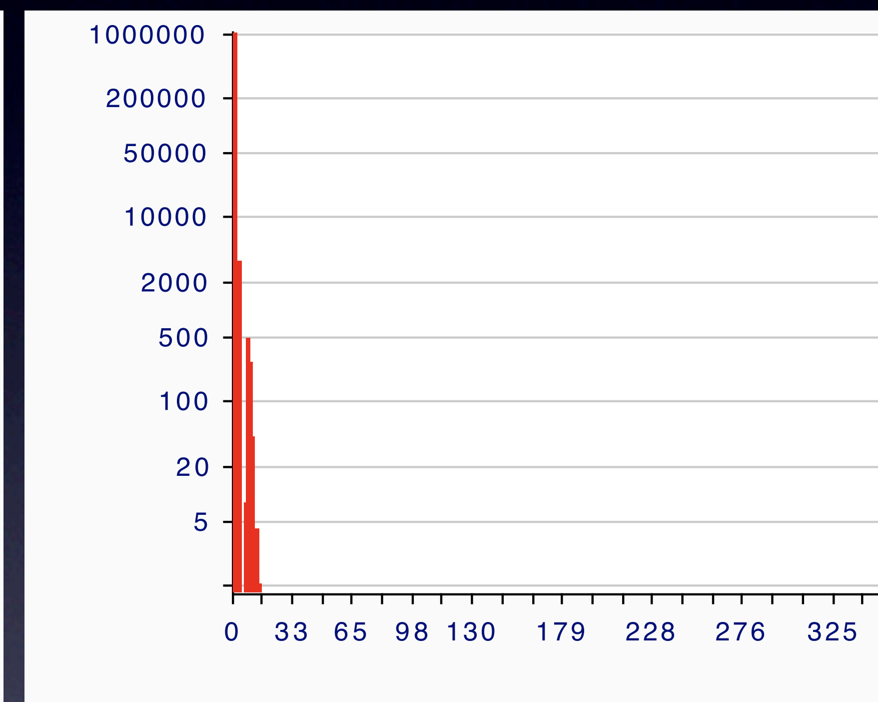
Execution Time Histograms



List

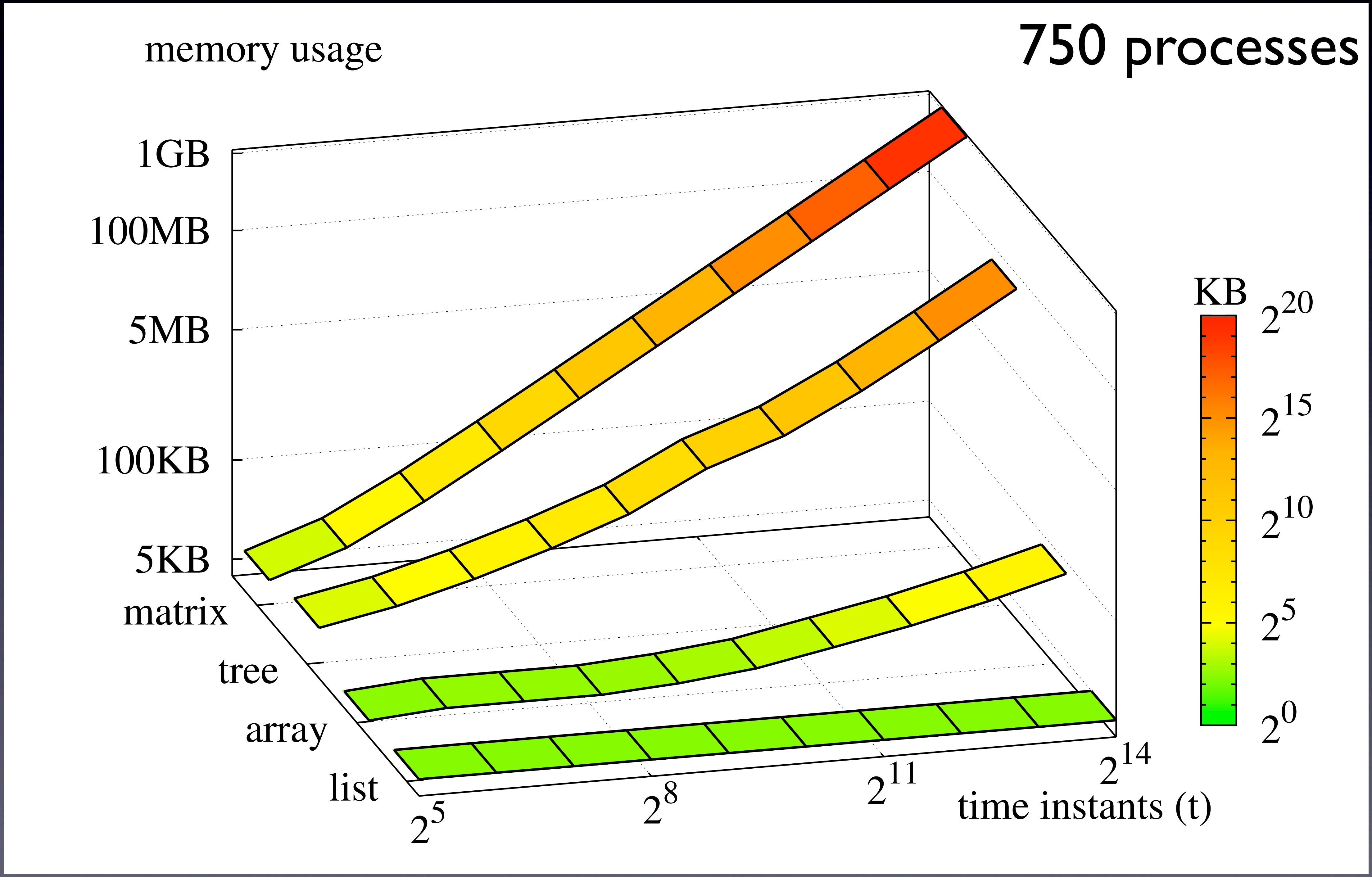


Array



Matrix

Memory Overhead



Outline

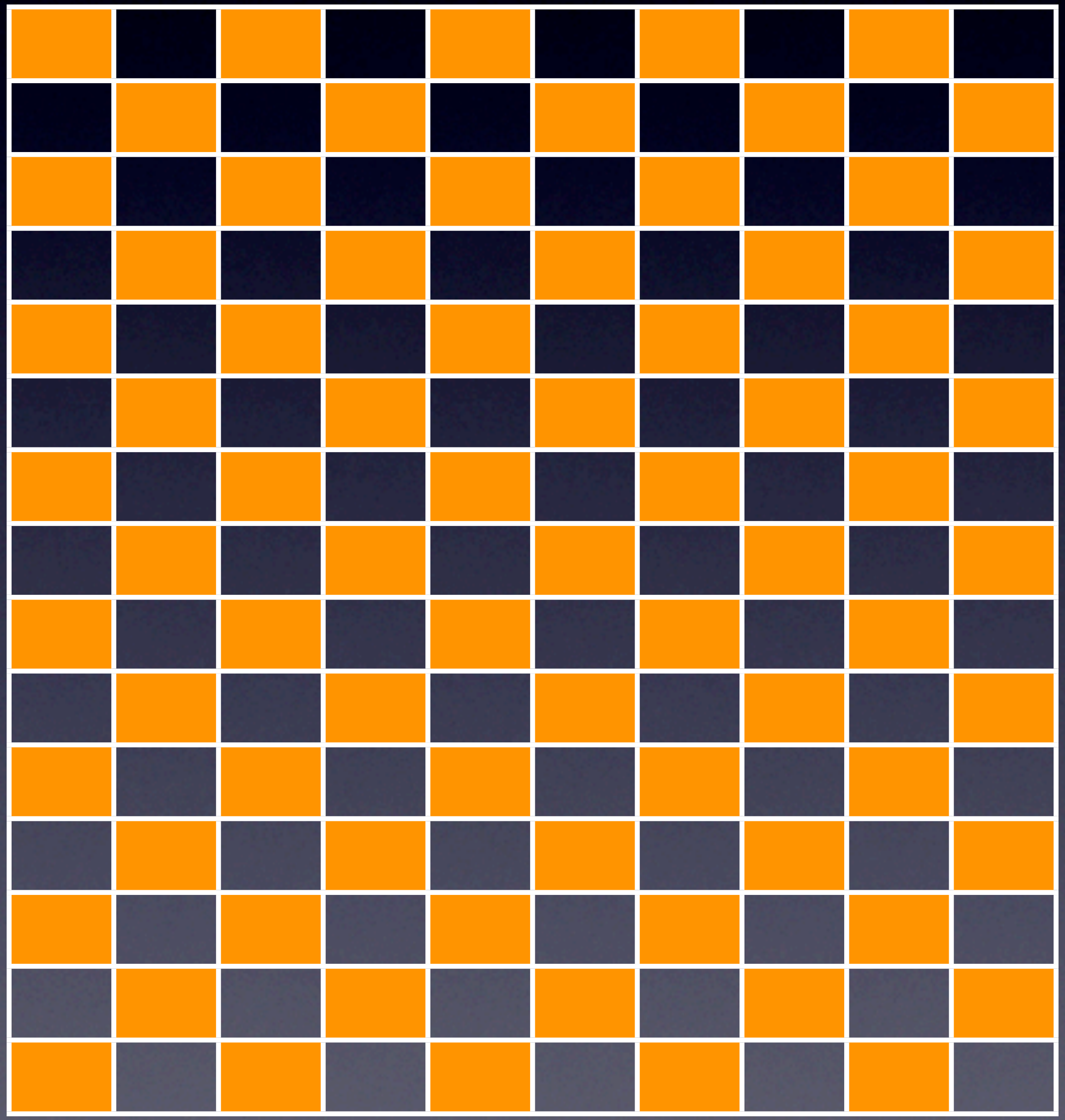
1. Time-Portable Programming
2. Tiptoe VM Scheduler
3. Tiptoe VM Memory Management

"Compact-Fit" [USENIX 2008]

- malloc(n) takes O(1)
- free(n) takes O(1) (or O(n) if compacting)
- access takes one indirection

- memory fragmentation is bounded and predictable in constant time

The Problem

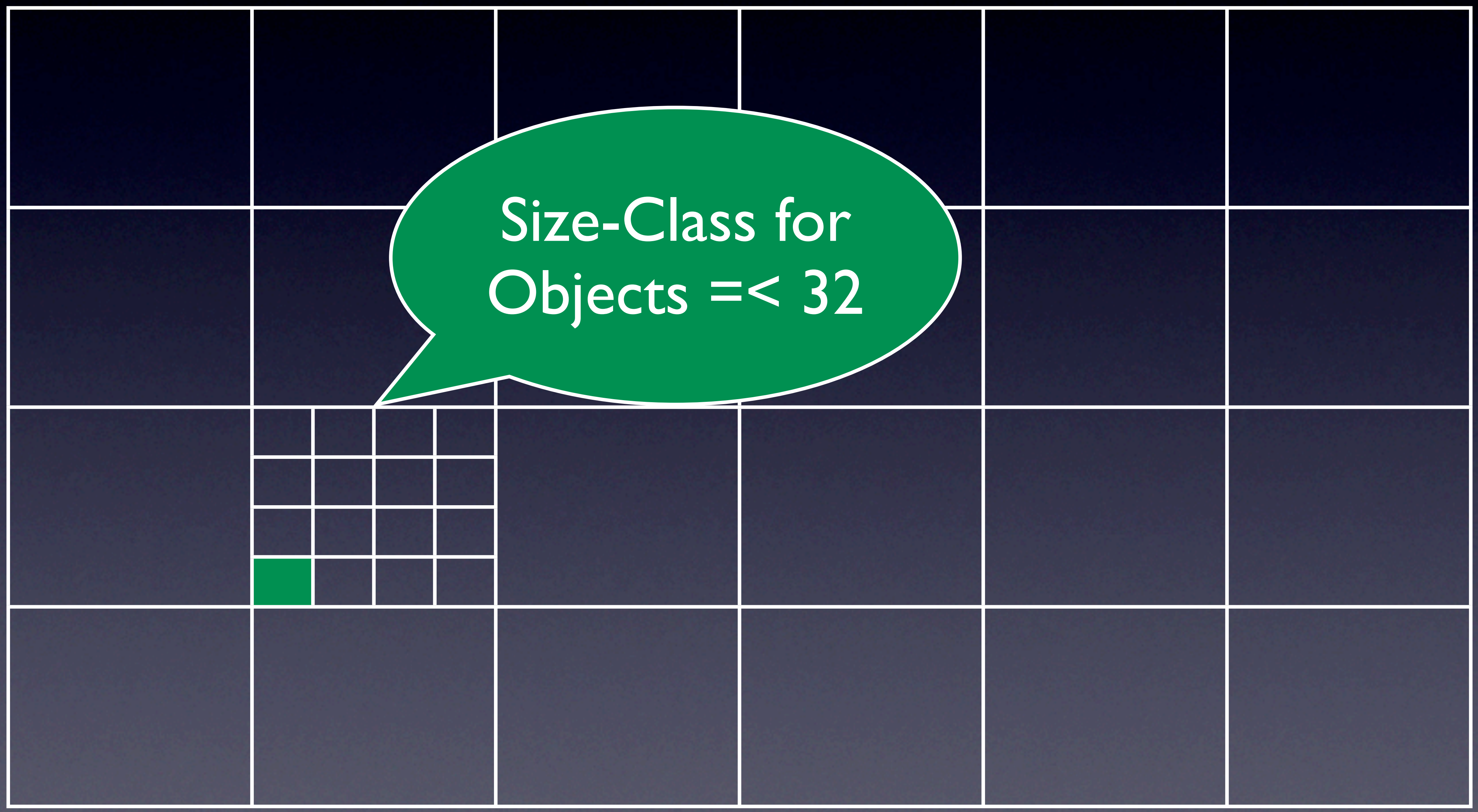


- Fragmentation
  - Compaction
- References
  - Abstract Space

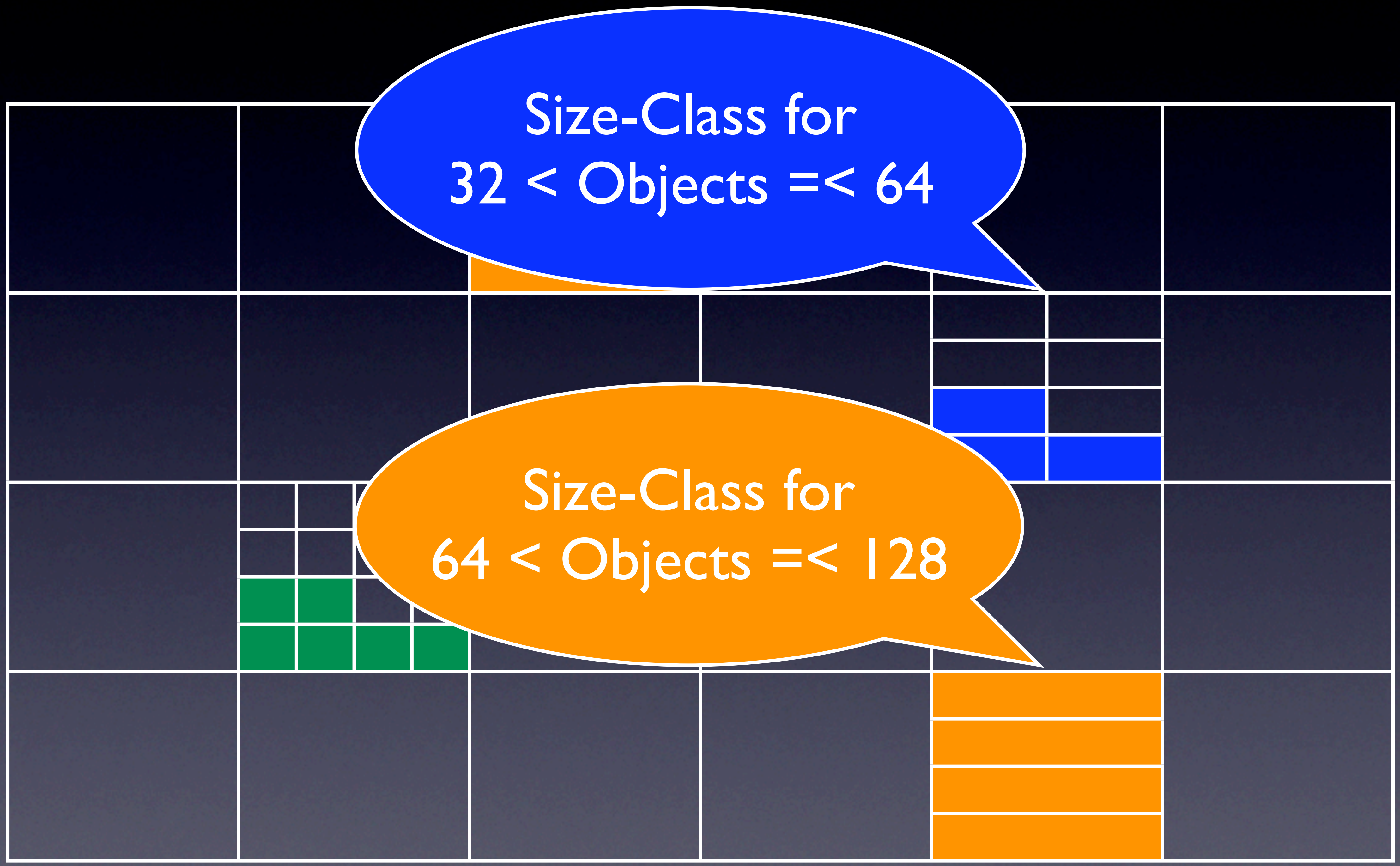
Partition Memory into Pages

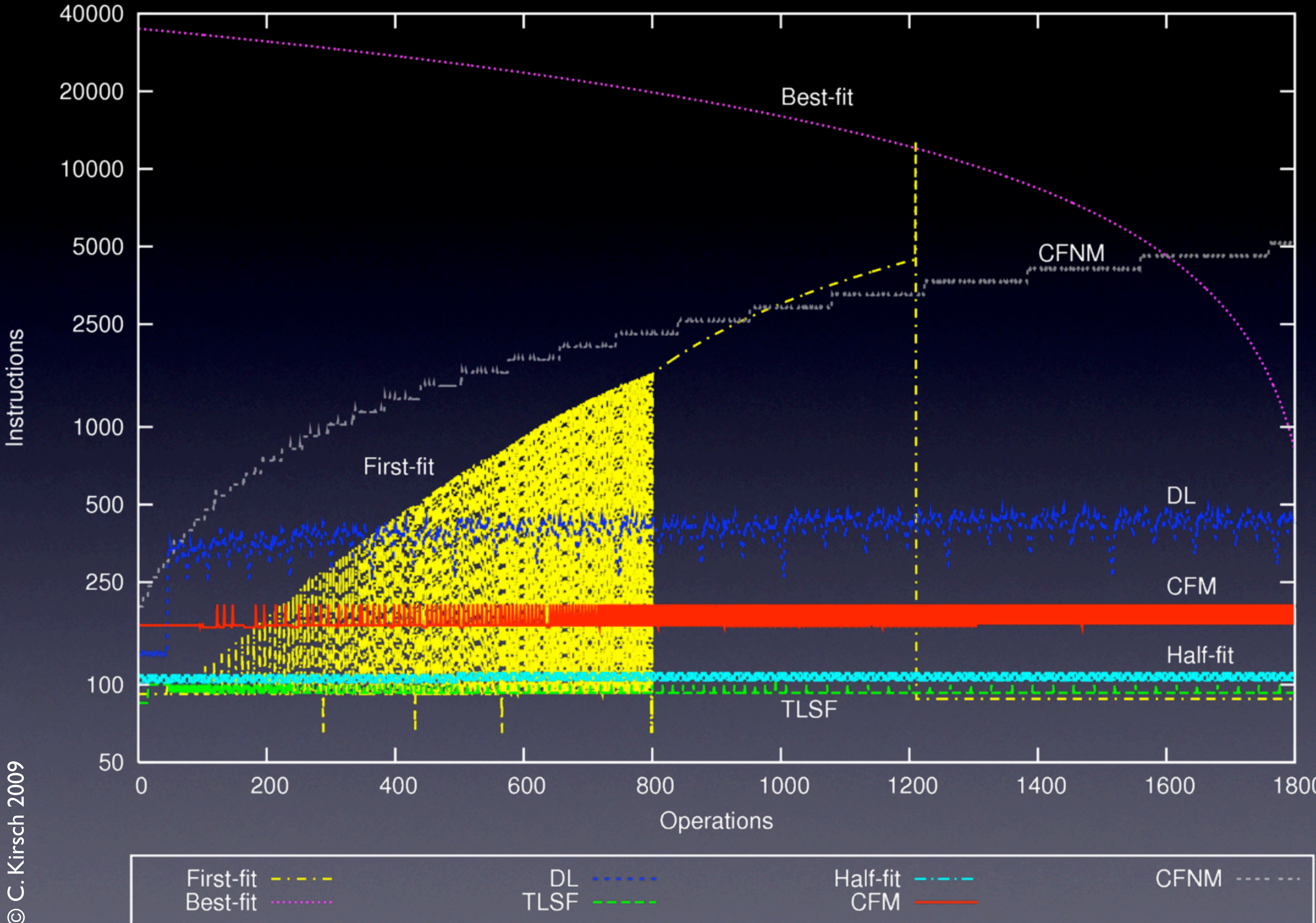
16KB	16KB	16KB	16KB	16KB	16KB
16KB	16KB	16KB	16KB	16KB	16KB
16KB	16KB	16KB	16KB	16KB	16KB
16KB	16KB	16KB	16KB	16KB	16KB

Partition Pages into Blocks



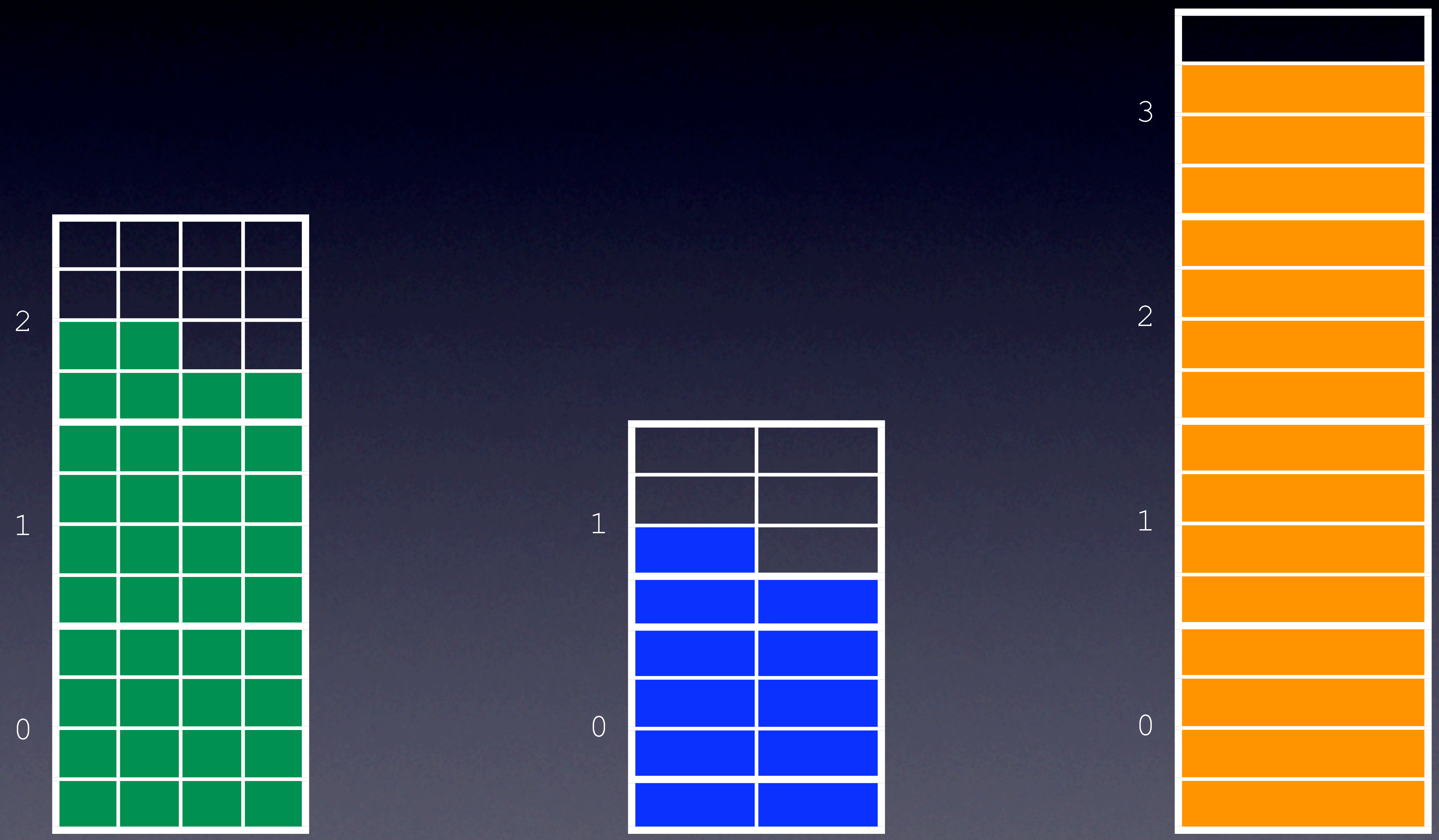
Size-Classes





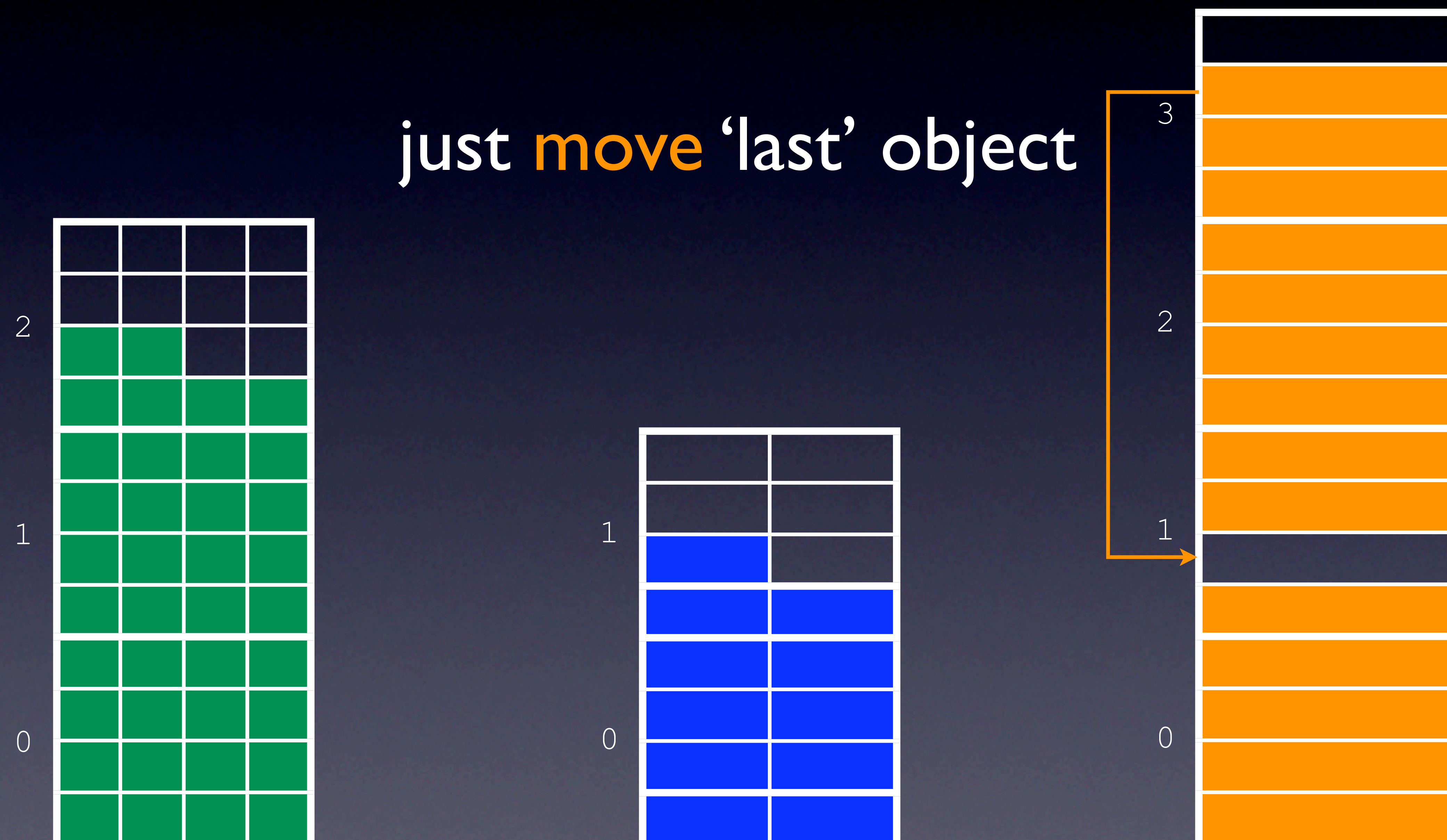
Instructions
40000
20000
10000
5000
2500
1000
500
250
100
50
0
200
400
600
800
1000
1200
1400
1600
1800
Operations
Best-fit
CFNM
First-fit
DL
CFM
Half-fit
TLSF
First-fit
Best-fit
DL
TLSF
Half-fit
CFM
CFNM

Invariant: Size-Class Compact

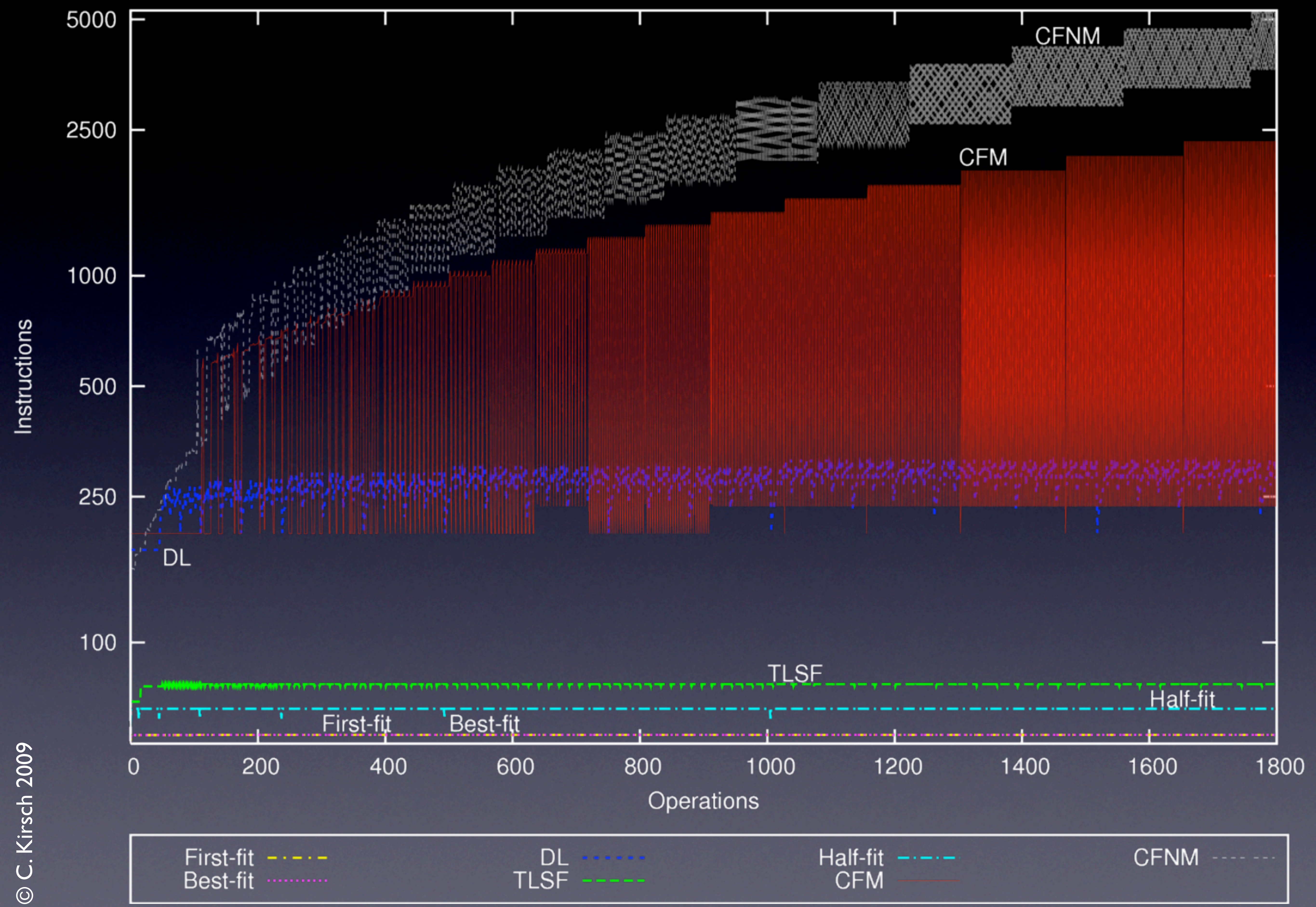


Objects =< 32 Objects =< 64 Objects =< 128

“Compact-Fit” (Bounded Compaction)

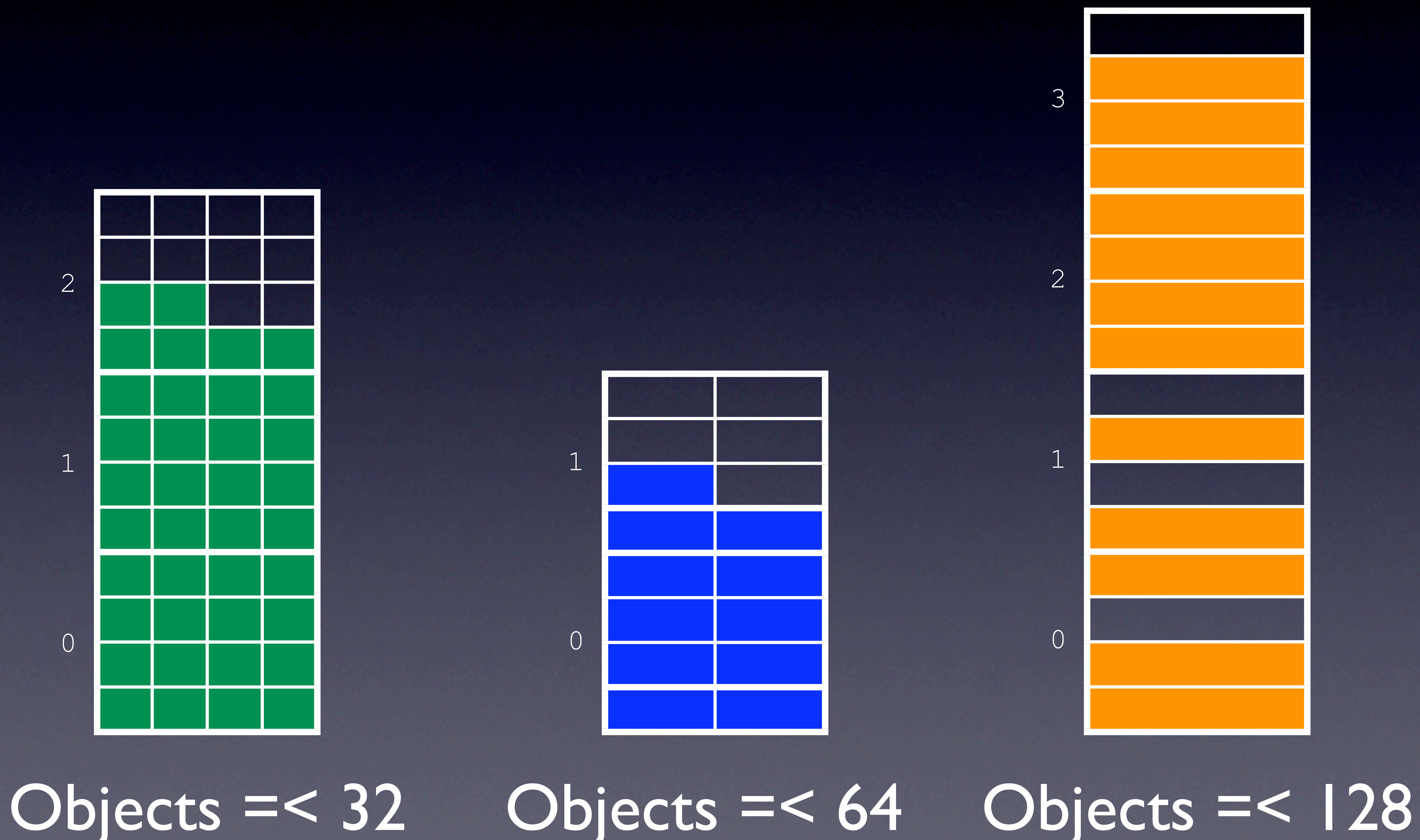


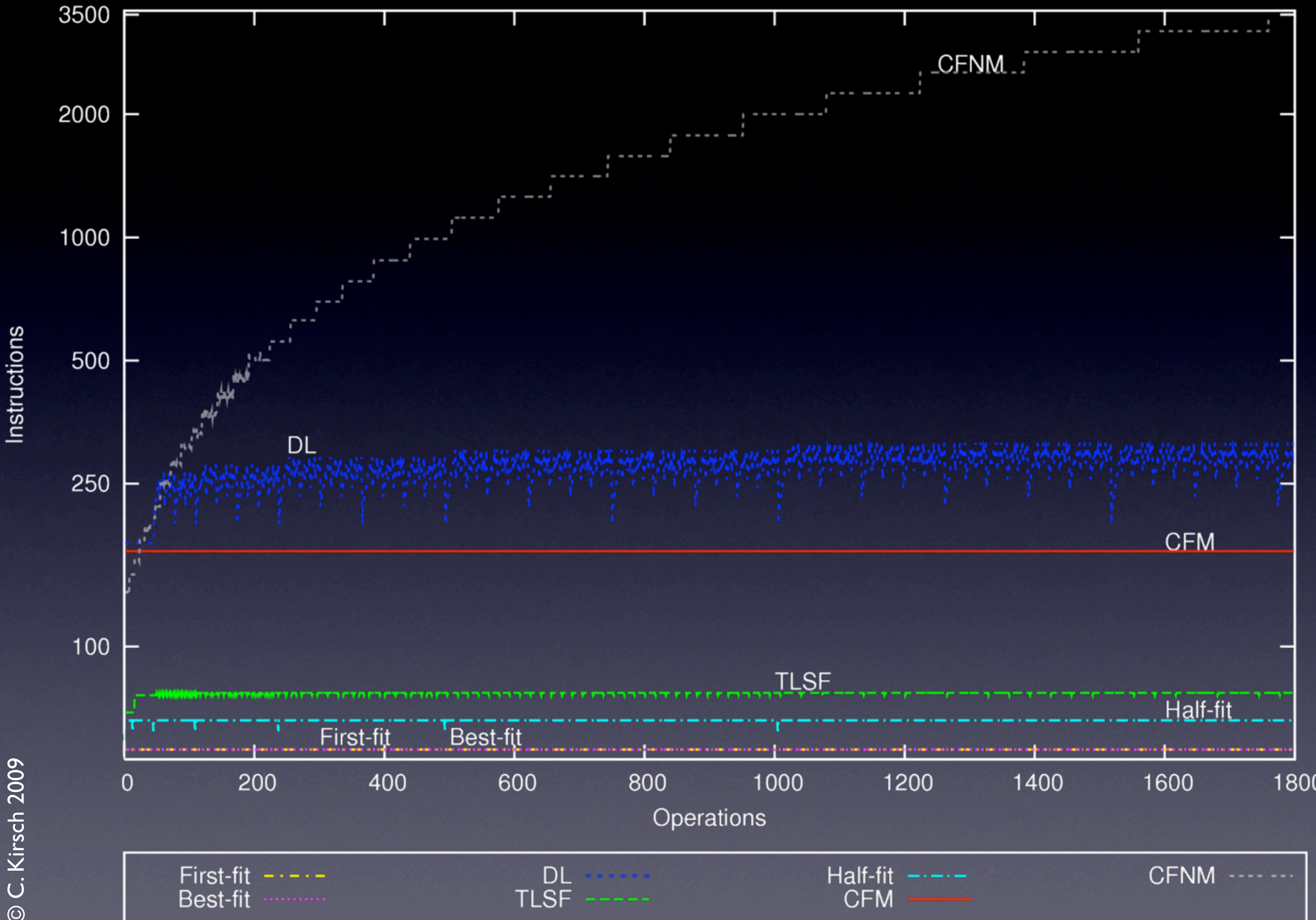
Objects =< 32 Objects =< 64 Objects =< 128



5000
2500
1000
500
250
100
Instructions
0
200
400
600
800
1000
1200
1400
1600
1800
Operations
CFNM
CFM
DL
TLSF
Half-fit
First-fit
Best-fit
First-fit
Best-fit
DL
TLSF
Half-fit
CFM
CFNM

Partial Compaction




CFNM
DL
CFM
TLSF
Half-fit
First-fit
Best-fit
Instructions
3500
2000
1000
500
250
100
0
200
400
600
800
1000
1200
1400
1600
1800
Operations
First-fit
Best-fit
DL
TLSF
Half-fit
CFM
CFNM

Current/Future Work

- Concurrent memory management
- Process management
- I/O subsystem


Thank you