oint Laboratory for Petascale Computation



The Actor Model and Multi-core Architectures

Emilio Francesquini

emilio@ime.usp.br

Alfredo Goldman

gold@ime.usp.br

Jean-François Méhaut

Jean-Francois.Mehaut@imag.fr

University of São Paulo University of Grenoble

8th Workshop of the INRIA-ANL-Illinois Joint Laboratory on Petascale Computing. Argonne, 11/19/2012

Outline

The Actor Model

Erlang

- Erlang's Actor Model Implementation
- Application Characteristics
- Ongoing Research

The Actor Model

The Actor Model

- Introduced by Hewitt et al in '73, followed by Agha in '86
 - No shared memory
 - Asynchronous message passing
- Strong integration with programming languages, not only as a library but as an integral part of it
- Actors are much like people. An actor
 - is born, spawns and kills other actors, and dies
 - Cannot read (or write) other actor's memories

The Actor Model Fine-Grained Parallelism

- The number of actors is typically much bigger than the number of processors.
 - In a typical simple application, the number of actors, is in the order of dozens or even hundreds
 - Charm++, for example, has a finer grain than MPI. The actor model is even more fine-grained
- This is reasonable since for most of the time they are just waiting for mail, i.e., inactive
- The lifespan of an actor, although variable, is usually short

Actor Model – CouchDB Application Characteristics



The Actor Model and Multi-core Architectures

Processing time (seconds)

Actor Model - CouchDB Actor Lifespan





Couch DB Communication Graph



Evolution of the actor's communication graph during the execution of the YCSB against CouchDB on a 24-core machine ()))

LINK to video.

Shows that, although not simple, there is a clearly defined structure of communication throughout the application's lifetime

There are clearly defined hub actors

The Actor Model and Multi-core Architectures

1/8 of the actual speed

Communication Graph

The communication graph is extremely dynamic
 It depends on the application and on the data

MapReduce is just one example of several possible sub-graph patterns

Some actors are clearly hubs, while others are just helpers

The Actor Model – Summing-up

Scales well in local and distributed systems

- Architecture agnostic programming
 - The runtime environment is responsible for all the necessary adaptations:
 - To efficiently use different machines (memory topologies, caches, processors, OSs, ...)
 - To conform to the needs of each individual application

Concurrency model used by Erlang, Scala, ...



Functional programming language

Created in the 80's by Ericsson



- Originally used for Ericsson's telephony routers
 High Availability Requirements
 Hot Code Replacement
- Available since 98 as Open Source Software

- Erlang Notable Use Cases
- Famous use cases:
 - Apache's CouchDB
 - WhatsApp (115K Messages/sec.)
 - Amazon's SimpleDB
 - Ericsson's AXD301 160Gbps ATM switch
 - Sim-Diasca, EDF's Discrete Event Simulator





- Concurrent (parallel and distributed) generic synchronous discrete-event simulation engine
- Fully implemented in Erlang
- Actively used by EDF and other partners
 - Used by the Smart Energy Supply Clever Project
 - The idea is to dynamically link energy supply and consumption
- Maintained by EDF R&D
- Released in 2010 by EDF R&D as free software (LGPL)

Erlang Virtual Machine

Runs on top of a custom built virtual machine

Application					
Erlang Virtual Machine					
Machine 1	Machine 2		Machine N		

- Only 3 concurrency constructs
 - Spawn Creates a new process
 - I Sends a message
 - receive Receives a message

Erlang Virtual Machine – Scheduler Architecture



- For each PU, one OS thread (called scheduler) is created
- There are specific VM options to determine how these schedulers are distributed throughout the available Pus
- Each scheduler has a run queue of actors ready to be executed

Erlang Virtual Machine – Memory Management

- Each actor in the system has its private heap
 - Small heap
 - No need to "stop the world" to do a garbage collect (at most only the actor being inspected is going to be stopped)
 - As the lifespan of each actor is usually short, most of them never experience garbage collections during their lives
- Message delivery is done by copying the message from the sending actor heap to the receiving actor heap

Erlang Virtual Machine

- The current implementation does not take into consideration
 The impact of the initial placement of each actor
 - Actor's migration
 - Cache
 - Bus contention
 - Memory topology

Erlang Actor Communication – NUMA Machine

Machine (6498)							
NUMANode P#0 (16GB)							
Socket P #0							
13 (24MB)							
L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)
L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)
owerne	Core P#1	Core P#2	oure P #o	Core P#8	Core P#9	Core P#10	Core P#11
Souther the Texas	Souther PA	Contra Pro	Columbia de la company	Solitabelor (** Tean capital	COBINE PA TOTO DES	Contra de Constant	Southing the Transpoort
						,	
NUMANode P#1	(1668)						
Socket P#1							
L3 (24MB)							
L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)
L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)
Core P#0	Core P#1	Core P#2	Core P#3	Core P#8	Core P#9	Core P#10	Core P#11
Southin Pa (magaar)	Solubiar (n. mingata)	Soundar pa Jose quar	Solahiar #s [run op.b.a]	Souther Ph. (magaz)	Schuber #s [ran model]	Southing Pa. (magaz)	Solahar # Transpag
NUMANode P#2	: (16GB)						
NUMANode P#2	:(1668)						
NUMANode P#2 Socket P#2 L3 (24MB)	:(1668)						
NUMANode P#2 Socket P#2 L3 (24MB) L2 (256KB)	(16GB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)
NUMANode P#2 Socket P#2 L3 (24MB) L2 (256KB) L1 (32KB)	(1668) L2 (256KB) L1 (32KB)	L2 (256KB) L1 (32KB)	L2 (256KB) L1 (32KB)	L2 (256KB) L1 (32KB)	L2 (256KB) L1 (32KB)	L2 (256KB) L1 (32KB)	L2 (256KB) L1 (32KB)
NUMANode P#2 Socket P#2 L3 (24MB) L2 (256KB) L1 (32KB) Core P#0	(1668) L2 (256KB) L1 (32KB) Core P#1	L2 (256K8) L1 (32K8) Core P#2	L2 (256KB) L1 (32KB) Core P#3	L2 (256KB) L1 (32KB) Core P#8	L2 (256KB) L1 (32KB) Core P#9	L2 (256KB) L1 (32KB) Core P#10	L2 (256KB) L1 (32KB) Core P#11
NUMANode P#2 Socket P#2 L3 (2448) L2 (2568) L1 (328) Core P#0 Socket P #0	(1668) (L2 (256K8) (L1 (32K8)) (Core P#1	L2 (256K8) L1 (32K8) Core P#2	L2 (256KB) L1 (32KB) Core P#3 Solidation to compare the second	L2 (256KB) L1 (32KB) Core P#8 Solders, manual	L2 (256KB) L1 (32KB) Core P#9 Selector rest	L2 (256KB) L1 (32KB) Core P#10 Settler A: Internet	L2 (256K8) L1 (32K8) Core P#11
NUMANode P#2 Socket P#2 L3 (2448) L2 (256K8) L1 (32K8) Core P#0	(1668) L2 (256K8) L1 (32K8) Core P#1	L2 (256KB) L1 (32KB) Core P#2	L2 (256(8)) L1 (32(8)) Core P #3	L2 (256KB) L1 (32KB) Core P #8	L2 (256KB) L1 (32KB) Core P #9	L2 (256(8)) L1 (32(8)) Core P+10 States to: massing	L2 (256KB) L1 (32KB) Core P#11
NUMANode P#2 L3 (2448) L2 (2568) L1 (328) Core P#0 (sector p)	(1669) (12 (256K8) (13 (25K8)) (13 (25K8)) (13 (25K8)) (13 (25K8)) (13 (25K8)) (13 (25K8)) (13 (25K8)) (13 (25K8)) (14 (25K8)	L2 (256K8) L1 (32K8) Core P#2	L2 (256(8)) L1 (32K8) Core P#3	L2 (2568) L1 (328) Core P 48	L2 (29(8)) L1 (32(8)) Core P H9	L2 (2568) L1 (328) Core P#10	L2 (256KB) L1 (25KB) Core P#11
NUMAIode P#2 Societ P#2 L3 (2448) L2 (256(8)) L1 (32(8)) Core P#0 (2007) (2007) NUMAIode P#3	(1568) L2(2568) L1(2588) Cere P1 (1569) (1569)	L1 (22669) L1 (3269) Core P #2 (###07)	L2 (256K8) L1 (32K8) Core P #3	L2 (256K8) L1 (32K8) Core P 49	L1 (25K8) L1 (32K8) Core P #9 Soldwirk Texator	L2 (256K8) L1 (32KB) Core P #10 Status In: Instatus	L2 (256(8) L1 (326) Core P911
NUMAIods P#2 Scidet P#2 L (2446) L (2568) L (2568) L (2568) NUMAIods P#3 L (2446)	(1568) (12(256(8)) (13(37K8)) (13(37K8)) (1568) (1568)	L2 (256K9) L1 (32K8) Core P#2 Core P#2	L2 (2568) L1 (3269) Core P#3	L2 (2560) L1 (3260) Core Pri (1997/10 (1997)	L2 (2560) L1 (258) Core 149	L2 (256KB) L1 (32KB) Cere 7#30	L2 (2568) L1 (328) Coc P911
NUMANode #2 Societ #42 L (2440) L (256(3) L (3280) Core #40 Core #40 Core #40 L (3240) L (256(3) L (240) L (240) L (240)	(1668) (12 (256(8)) (1 (32(8)) (1 (32(8))) (1 (32(8)) (1 (32(8))) (1 (32(8)))) (1 (32(8)))) (1 (32(8)))) (1 (32(8)))) (1 (32(8	L2 (256K9) L1 (35K9) Core P#2 (00078 (0007) L2 (256K9)	L2 (2560) L1 (3260) Core 7#3 (ministry (ministry)) L2 (2560)	L2 (2560) L1 (326) Core P#8 Core P#8 L2 (259(0))	L2 (256K) L1 (35K) Core 799 (MARTIN L1 (35K) L2 (256K)	L2 (256K8) L1 (32K8) Core P#10 Filler (1256K8) L2 (256K8)	L2 (2560) L1 (326) Core P01 Core P01 L2 (2560)
NUMANode P2 Sodat P42 L3 (2446) L2 (256(3)) L1 (3266) Sodat P45 L3 (2446) L3 (2466) L3 (246	(1668) (12 (2566)) (1 (326)) (1698) (1698) (12 (2266)) (1 (326)) (1 (326))	L2 (256(8)) L1 (32K8) Core P#2 (5557 (11) L2 (256(8)) L1 (32K8)	L2 (259(8)) L1 (32(8)) Core 7#3 Status Jr. (1990) L2 (259(8)) L1 (32(8))	L2 (259(8)) L1 (3258) Core P #8 (5557) (21) L2 (229(8)) L1 (328)	L2 (259(8)) L1 (32(8)) Core P #9 (minima in the second	L2 (256K8) L3 (35K8) Cece P #10 1000 h. (1000) L2 (256K8) L2 (256K8) L1 (32K8)	L (2566) L (326) Core P01 Core P01 L (326) L (2566) L (326)
NUMANode #2 Socket #42 L3 (2446) L2 (256(3), L1 (32K6) Core #40 Core #40 Core #40 L3 (2446) L3 (2466) L3 (2466)	(1668) (12 (2566)) (1 (2566)) (2 (2566)) (2 (2566)) (2 (2 (2566))) (2 (2 (2566))) (2 (2 (2566))) (2 (2 (2 (2566))) (2 (2 (2 (2 (2566)))) (2 (2 (2 (2 (2 (2 (2 (2 (2 (2 (2 (2 (2 (L2 (256(8)) L1 (226) Core P#2 Market (256(9)) L2 (256(9)) L1 (22(8)) L1 (22(8)) Core P#2	L2 (2566) L1 (3269) Core P#3 Status Fr. (1990) L2 (2560) L1 (328) L2 (2560)	L2 (259(0) L1 (3289) Core P#8 L2 (259(0) L1 (3280) L1 (3280) L2 (259(0)	L2 (254K)) L1 (25K) Core P#9 Stature many L2 (254K)) L1 (25K) L1 (25K)	L2 (256(8) L3 (35/8) Core P #10 5660 P. 1000 5660 P. 1000 L2 (256(8) L2 (256(8) L2 (256(8)) L2 (256(8	L (2566) L (326) Core P#1 Fotors mini- L (2569) L (2269) L (2329) L (2329)

- Message exchange between two actors on the same NUMA node
- If the message is small enough to fit the caches, it can be sent using them, otherwise it ends up using the local node memory

Erlang Actor Communication – NUMA Machine

NUMANode P#							
ocket P#0							
L3 (24MB)							
L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)
L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)
core P#o	Core P#1	Core P#2	Core P#3	Core P#8	Core P#9	Core P#10	Core P#11
		Silehier #*				Solitabelar (m. 1995)	Solution (m.
NUMANode P#1	(16GB)						
Socket P#1							
L3 (24MB)							
L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)	L2 (256KB)
L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)	L1 (32KB)
			C D #2	Core P#8	Core P#9	Core P#10	Coro P#11
Core P#0	Core P#1	Core P#2	Core F#5				COLET WAL
Core P#0	Sitehir fa teripti	Sitchir fr	Sidder An angel	Singhing the Transport	Silishir (n. 1444)	Silehir (n. 1000)	Sishir fa
Core P#0	Core P#1	Core P #2		Statics in Transfer	Sitchir (* Tranipite)	Sölshör (* Janapos)	Sitting in The state
Core P#0	Core P#1	Core P#2	Sobler In mappe				Silehier fin Innippe
Core P#0	Core P#1	Core P#2	Socie in an and a		Söllskir (*)	Solithiar (n. Innajači	Solithiar in an appart
Core P#0	Core P#1	Sotther is images				Stellule de Terrande	District for an and a second s
Core P#0 Stable # Imperial NUMANode Socket P#2 L3 (24MB)	Core P#1			Sidde In Transford	Solehice for the test state	Soldware for an unit	
Core P#0 Status 14 Installed NUMANode Socket P#2 L3 (24MB) L2 (256KB)	Core P #1 Sether fsstates (16GB)	Core P#2	Coorras Seleta (256KB)	L2 (256KB)	District to	L2 (256KB)	Sottier Provide Sottier Provid
Core P#0 Southur 14 (2014) NUMANodo Socket P#2 L3 (24MB) L2 (256KB) L1 (32KB)	Core P #1 SetMar 5 L2 (256KB) L1 (32KB)	Core P#2 SetAble 16 L2 (256KB) L1 (32KB)	L2 (256KB)	L2 (256KB) L1 (32KB)	L2 (256KB) L1 (32KB)	L2 (256KB) L1 (32KB)	L2 (256KB) L1 (32KB)
Core P#0 Smbhr fr Immedia Socket P#2 L3 (24MB) L2 (256KB) L1 (32KB) Core P#0	L2 (256KB) L1 (32KB) Core P#1	L2 (256KB) L1 (32KB) Core P#2	L2 (256KB) L1 (32KB) Core P#3	L2 (256K8) L1 (32K8) Core P#8	L2 (256KB) L1 (32KB) Core P#9	L2 (256KB) L1 (32KB) C ore P#10	L2 (256KB) L1 (32KB) Core P#11
Core P #0 Steller in INUMANIde Socket P #2 L3 (24MB) L2 (256KB) L1 (32KB) Core P #0 Core P #0	L2 (256KB) L1 (32KB)	L2 (256KB) L1 (32KB)	L2 (256KB) L1 (32KB)	(25642 / 1000 /	L2 (256KB) L1 (32KB) Core P#9	L2 (256KB) L1 (32KB) Core P #10	L2 (256KB) L1 (32KB) Core P#11
Core P #0 Stabur () Stabur	L2 (2568) L2 (2568) L3 (328) Core Pal (motors man)	L2 (25K8) L2 (25K8) L1 (32K8) Core P#2 fmlore 1	L2 (256(8) L1 (2X8) Core P#3 State h (11)	L2 (256K8) L1 (226K8) Core P#8 More Lassi	L2 (25K8) L1 (25K8) Core P#9 Online Internet	L2 (256K8) L1 (25K8) Core P#10 Core P#10	L2 (256KB) L1 (32KB)
Core P #0 Stabin () Stabin () S	L2 (2568) L2 (2568) Core P=1 (1528)	L2 (25K8) L2 (25K8) L1 (32K8) Core P # 1 future to 1	L2 (256K8) L2 (256K8) L1 (32KB)	L2 (256K8) L1 (32K8) Core P #8	(index p. index) (index p. index) (index p. index) (index p. index) (index p. index) (index p. index)	L2 (256(8)) L1 (3268) Core P #10 Solder In Testing	(11) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2
Core P #0 Solder P #2 Solder P #2 L3 (24MB) L2 (256KB) L1 (32KB) Core P #0 Solder P #0 S	L2 (2568) L1 (328) Core P41 (1568)	L2 (25K8) L2 (25K8) L1 (32K8) Core P#2 Mathema (25K8)	L2 (25KB) L2 (25KB) L1 (32KB)	L2 (256K8) L1 (32K8) Core P #9	UL (25KR) LL (25KR) Core P #9 Solder Tr. Image J	L1 (2540) L1 (3240) Core P #10 Solida Tr. Image	L2 (256KB) L1 (32KB) Core P#11
Core P #0 Socket P #2 L3 (24MB) L2 (256KB) L1 (32KB) Core P #0 Socket P #3 NUMANode P #3	L2 (2568) L2 (2568) L2 (2568) L3 (2588) Core Pil (2568)	L2(2560) L1(2560) L1(3260)	L2 (2568) L2 (2568) L3 (3268)	L2 (256(8) L1 (256(8) L1 (3268) Core P#0 1		L2 (256K) L1 (256K) D1 (32K) Core (#20 Main /n (main)	L2 (256KB) L2 (256KB) L1 (32KB) Core P#11
Core P #0	L2 (2568) L2 (2568) L1 (328) Core P#1 (1668)	L2(2560) L2(2560) L1(3260)		L2 (256K8) L1 (256K8) L1 (32K8) Core P#0 +	L2 (256(8)) L1 (326) Core P#9 mater #:	L2 (256K) L1 (256K) L1 (32KB)	L2 (256KB) L2 (256KB) L1 (32KB) Core P#11
Core P #0	L2 (2566) L2 (2566) L1 (326) L2 (2566) L2 (2506) L2 (2506)	L2 (256(8) L1 (256(8) L1 (3268) Core #2 L2 (259(8)		L2 (256(8) L1 (32K8) Cere P#0 L2 (256(9) L1 (22K8)	L2 (256(8) L1 (32(8) Core 7#3 mbb/h. <u>busis</u>)	L2 (254K8) L1 (32K8) Core P #10 million to total	L2 (256KB) L1 (32KB) L2 (256KB) L1 (32KB) Core P#11 L2 (256KB)
Core P #0 South 7 200 South 7 20 South	L2 (2968) L1 (2569) L2 (2968) L1 (3255) Core #1 (1668) L1 (3256) L1 (3256)	L2 (256(8) L1 (258(8) L1 (328) Core P #2 L2 (259(8) L1 (328) L2 (259(8)) L1 (328)	L2 (256K8) L1 (256K8) L1 (32K8) Core P#2 Series n	L2 (256(8)) L1 (32K8) Cere 948 L2 (256(8)) L1 (32K8) L1 (32K8)	L2 (256(8) L2 (256(8) (L1 (328)) Core P #9 (minimized in the second	L2 (256(8)) L1 (328) Core P #10 Solida Tr. Instein L2 (256(8)) L2 (256(8)) L1 (328)	L2 (256KB) L2 (256KB) L1 (92KB) L2 (256KB) L1 (92KB) L2 (256KB) L1 (92KB)

Message exchange between two actors on distinct NUMA nodes

 Communication through the node's interconnect

Actor Communication Time - Message sizes from 1 B to 65KB - idkonn - 4 sockets - 6 cores per socket. Two cores/L2, Six cores/L3



Ongoing Research

Actor Mapping to Multicore Machines



- How to distribute the actors throughout the schedulers, i.e., find a good mapping of the communication graph to the machine
 - To minimize the makespan of the application
 - To take advantage of the hierarchical memory for communication efficiency

Machine (64GB UMANode P#0 (16GB Socket P#0 L3 (24MB) 1.2 (256KB) L2 (256KB) L1 (32KB) L1 (32KB) 1 (32KB) L1 (32KB) 11 (32KB) L1 (32KB) 11 (32KB) L1 (32KB) Core P#11 Core P#0 Core P#: Core P#2 Core P#3 ore P#8 Core P#9 Core P#10 PU P#0 PU P#4 PU P#8 PU P#12 PU P#16 PU P#20 PU P#24 PU P#28

NUMANode P#1 (1668)	
Socket P#1	-
L3 (24MB)	٦
L2 (256KB)]
L1 (32KB)]
Core P#0 Core P#1 Core P#2 Core P#3 Core P#3 Core P#1 Core P#1 PU P#23 Core P#3 Core P#3 <td< td=""><td></td></td<>	

NUMANode P#2 (16GB)	
Socket P#2 L3 (24MB)	
L2 (256KB) L2 (256KB) L2 (256KB)	L2 (256KB) L2 (256KB) L2 (256KB)
L1 (32KB) L1 (32KB) L1 (32KB) L1 (32KB)	L1 (32KB) L1 (32KB) L1 (32KB) L1 (32KB)
Core P#0 Core P#1 Core P#2 Core P#3 PU P#2 PU P#6 PU P#10 Core P#3	Core P#8 Core P#9 Core P#10 Core P#11 PU P#18 PU P#22 PU P#26 Core P#11

NUMANode P#3 (1668)
Socket P#3
L3 (24MB)
L2 (256K8) L2 (256
L1 (32KB)
Core P#0 Core P#1 Core P#3 Core P#3 Core P#3 Core P#1 Core P#3

Initial Actor Placement

- Most of the actors have a short life
 - initial placement importance
 - We could use strategies like BubbleSched for this since the decision must be done fast
- Is the father of the actor a good indicator of its behavior?
- We do not know the actual behavior of the actor. Can we predict it based on the past?



Default Strategy – 3,030 Actors ~4,560 migrations

The Actor Model and Multi-core Architectures

24

Scheduler 1				
Scheduler 2				
Scheduler 3				
Scheduler 4			<u> </u>	
Scheduler 5				
Scheduler 6			and the second s	
Scheduler 7				
Scheduler 8				
Scheduler 9			and the second se	
Scheduler 10			9	
Scheduler 11				
Scheduler 12				
Scheduler 13		· · · · · · · · · · · · · · · · · · ·		
Scheduler 14				
Scheduler 15				
Scheduler 16				
Scheduler 17		· · · · · · · · · · · · · · · · · · ·		
Scheduler 18			·····	
Scheduler 19				
Scheduler 20				
Scheduler 21				
Scheduler 22		····•		
Scheduler 23				
Scheduler 24				
Scheduler 25				
Scheduler 26				
Scheduler 27				
Scheduler 28				
Scheduler 29				
Scheduler 30				
Scheduler 31				
Scheduler 32				000
Check Balance		<u> </u>		

Default Strategy – 3,030 Actors ~4,560 migrations

The Actor Model and Multi-core Architectures

25

		Contraction of the second s		
Scheduler 1				
Scheduler 2	T THE REAL PROPERTY AND A REAL			
Scheduler 3				
Scheduler 4				
Scheduler 5			X	
Scheduler 6			Y	
Scheduler 7				
Scheduler 8			e e e e e e e e e e e e e e e e e e e	
Scheduler 9				
Scheduler 10				
Scheduler 11				
Scheduler 12				
Scheduler 13				
Scheduler 14				
Scheduler 15				
Scheduler 16				
Scheduler 17				
Scheduler 18				
Scheduler 19				
Scheduler 20				
Scheduler 21				
Scheduler 22				
Scheduler 23				
Scheduler 24				
Scheduler 25				
Scheduler 26				
Scheduler 27				
Scheduler 28				
Scheduler 29				
Scheduler 30				
Scheduler 31				
Scheduler 32			000	
Check Balance		0 00	000	
	Round Robin Strategy – 3,030 Actor	ſS		
	~610 midrations		00	
			26	

Scheduler 1		
Scheduler 2		
Scheduler 3		
Scheduler 4		
Scheduler 5		
Scheduler 6		
Scheduler 7		
Scheduler 8		
Scheduler 9		
Scheduler 10		
Scheduler 11		
Scheduler 12		
Scheduler 13		
Scheduler 14		
Scheduler 15		
Scheduler 16		
Scheduler 17		
Scheduler 18		
Scheduler 19		
Scheduler 20		
Scheduler 21		
Scheduler 22		
Scheduler 23		
Scheduler 24		
Scheduler 25		
Scheduler 26		
Scheduler 27		
Scheduler 28		
Scheduler 29		
Scheduler 30		
Scheduler 31		
Scheduler 32		
Check Balance		100
	Round Robin Strategy – 3,030 Actors	
	~biu midrations	07
		21

Migration Count X Initial Placement Strategy

Strategy	Avg. of 30 Executions
Default	4,560
Round Robin	573
Random	1,244

3,030 Actors ~20 seconds of execution time

Initial Placement and Execution Times

Execution Time Difference



Ongoing Research

- Actor migration
 - Only makes sense for the long lived actors
 - The analysis of the communication graph (highly dynamic) should be taken into consideration
 - The procurement of the communication graph is trivial, doing it fast is not
 - Iterative and continuous process
 - Should be automatic
 - Trying to choose the best placement based on the current and historical data?

Work in progress

Initial Actor Placement

- Actors in general have a short life, therefore the initial placement decision must be fast
- Idea: Implement something similar to BubbleSched

Actor Migration

- Establish migration paths for each actor to minimize the makespan based on the communication graph and the machine topology
- Idea: Hub actors, if placed together, might saturate the bus. Place the hub nodes as far as possible to increase performance
- Actor Pinning
 - Automatic for hub actors, and application developers might have insights as to the best placement of each actor
 - Idea: Implement an interface that allows the application developer to specify where the actors should be placed

Thank you!