Grid programming with components:
an advanced COMPonent platform
for an effective invisible grid

GridCOMP
Effective Components for the Grids

Adaptative Behavior with GCM

Marco Aldinucci, M. Danelutto, S. Campa
University of Pisa, Italy

D. Laforenza, N. Tonellotto, P.Dazzi
ISTI-CNR, Italy

October 31th, 2007
Beijing, China
北京 - 中华人民共和国

Outline

- Motivation
  - why adaptive and autonomic management
  - why skeletons
- Behavioural Skeletons
  - parametric composite component with management
  - functional and non-functional description
  - families of behavioural skeletons
- GCM implementation
  - some hints today, much more tomorrow
    - Nicola Tonellotto and Patrizio Dazzi talk at "ProActive and GCM Tutorial and Hands-On Grid Programming" (Thursday 15,30-16,30)
  - preliminary experiments and performances

CGM model key points

- Hierarchic model
  - Expressiveness
  - Structured composition
- Interactions among components
  - Collective/group
  - Configurable/programmable
  - Not only RPC, but also stream/event
- NF aspects and QoS control
  - Autonomic computing paradigm

Why Autonomic Computing

- // programming & the grid
    - concurrency exploitation, concurrent activities set up, mapping/scheduling, communication/synchronisation handling and data allocation, ...
    - manage resources heterogeneity and unreliability, networks latency and bandwidth unsteadiness, resources topology and availability changes,

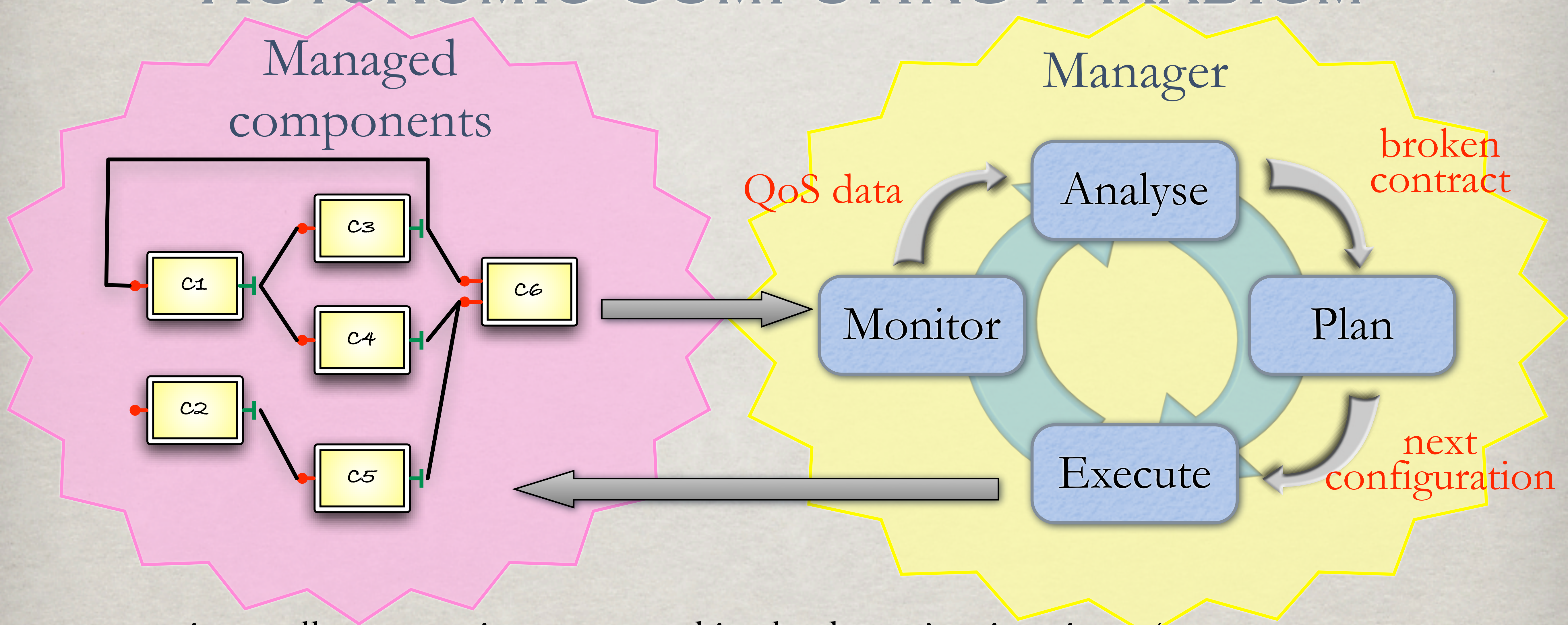
... and a non trivial QoS for **applications**

not easy leveraging only on middleware

our approach:

high-level methodologies + tools

Autonomic Computing paradigm



- monitor: collect execution stats: machine load, service time, input/output queues lengths, ...
- analyse: instantiate performance models with monitored data, detect broken contract, in and in the case try to detect the cause of the problem
- plan: select a (predefined or user defined) strategy to re-convey the contract to validity. The strategy is actually a "program" using execute API
- execute: leverage on mechanism to apply the plan

Why skeletons 1/2

- Management is difficult
  - Application change along time (ADL not enough)
  - How "describe" functional, non-functional features and their inter-relations?
  - The low-level programming of component and its management is simply too complex
- Component reuse is already a problem
  - Specialising component yet more with management strategy would just worsen the problem
  - Especially if the component should be reverse engineered to be used (its behaviour may change along the run)

Why skeletons 2/2

- Skeletons represent patterns of parallel computations (expressed in GCM as graphs of components)
- Exploit the inherent skeleton semantics
  - thus, restrict the general case of skeleton assembly
  - graph of any component ➠ parametric networks of components exhibiting a given property
  - enough general to enable reuse
  - enough restricted to predetermine management strategies
- Can be enforced with additional requirements
  - E.g.: Any adaptation does not change the functional semantics

Behavioural Skeletons idea

- Represent an evolution of the algorithmic skeleton concept for component management
    - abstract parametric paradigms of component assembly
    - specialized to solve one or more management goals
        - self-configuration/optimization/healing/protection.
- Are higher-order components
- Are not exclusive
    - can be composed with non-skeletal assemblies via standard components connectors
        - overcome a classic limitation of skeletal systems

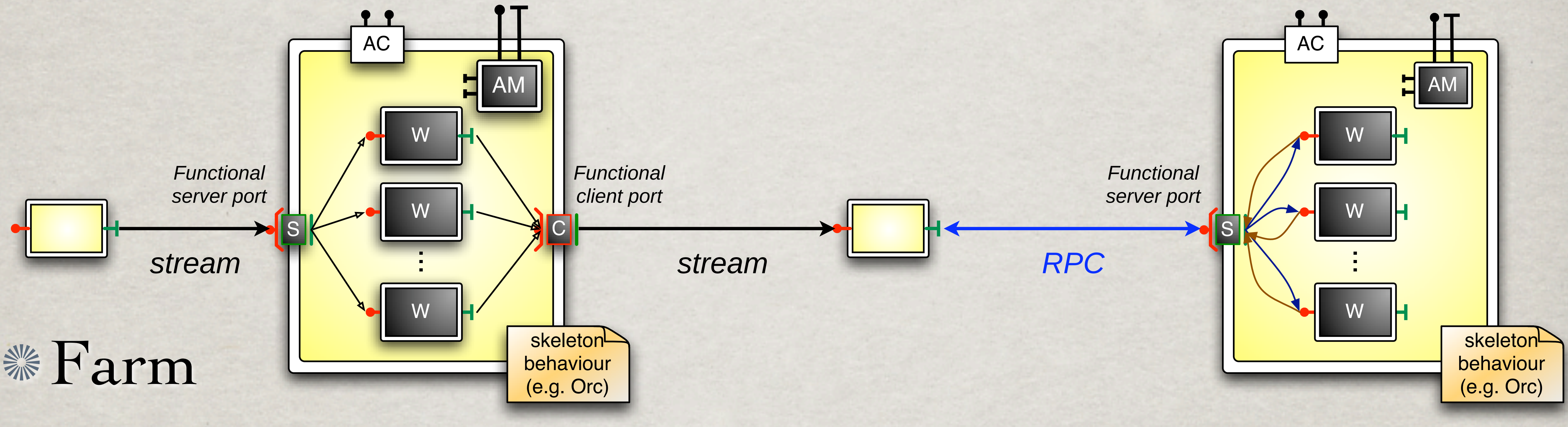
Behavioural Skeletons proprieties

- Expose a description of its functional behaviour
- Establish a parametric orchestration schema of inner components
- May carry constraints that inner components are required to comply with
- May carry a number of pre-defined plans aiming to cope with a given self-management goal
- Carry an implementation (they are factories)

Be-Skeletons families

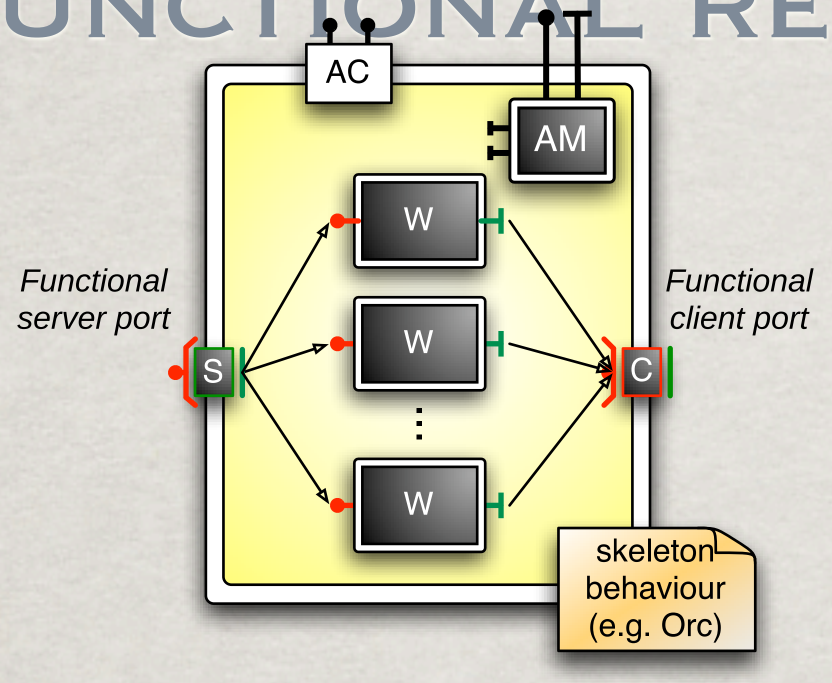
- Functional Replication
  - Farm/parameter sweep (self-optimization)
  - Simple Data-Parallel (self-configuring map-reduce)
  - Active/Passive Replication (self-healing)
- Proxy
  - Pipeline (coupled self-protecting proxies)
- Wrappers
  - Facade (self-protection)
- Many others can be borrowed from Design Patterns

Functional replication



- Farm
  - S = unicast, C = from_any, W = stateless inner component
- Data Parallel
  - S = scatter, C = gather, W = stateless inner component
- Fault-tolerant Active Replication
  - S = broadcast, C = get_one_in_a_set, W= stateless inner ...
- ...

Functional replication

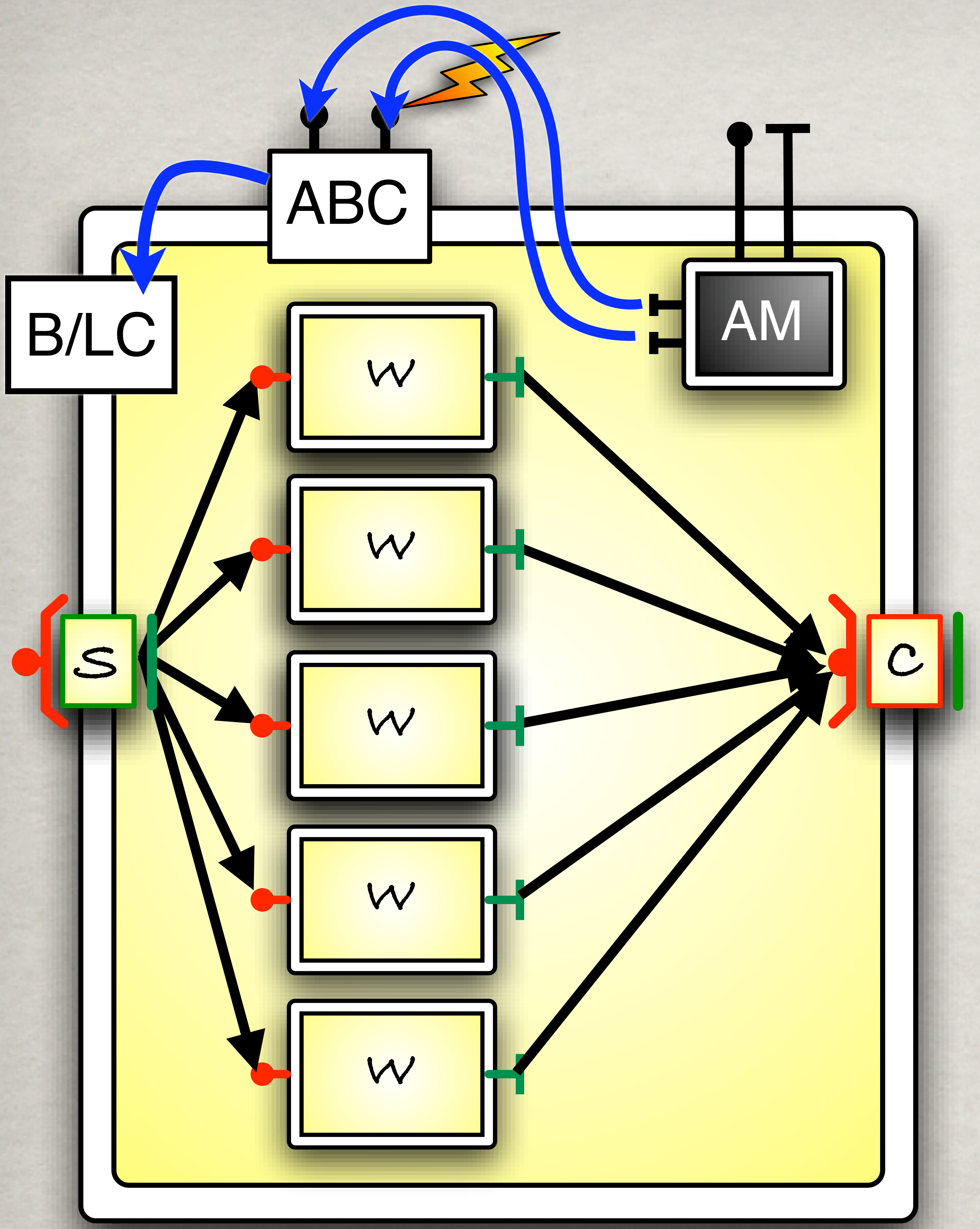


Functional behaviour description (orchestration)

$$system(data, S, G, W, in, out, N) \triangleq$$
$$S(data, in) \mid (\mid i : 1 \leq i \leq N : W_i(in_i, out_i)) \mid C(out)$$
$$W_i(in_i, out_i) \triangleq$$
$$in_i.get > tk > process(tk) > r > (out_i.put(r) \mid W_i(in_i, out_i))$$

- Meant to parametrically expose all allowed adaptation
  - Any AM policy that does not change this semantics is *correct*
  - As an example changing ***i*** in this schema is correct
  - Functional semantics is invariant from ***i***, non-functional one is not (and changing ***i*** means changing the number of Ws for self-* purposes

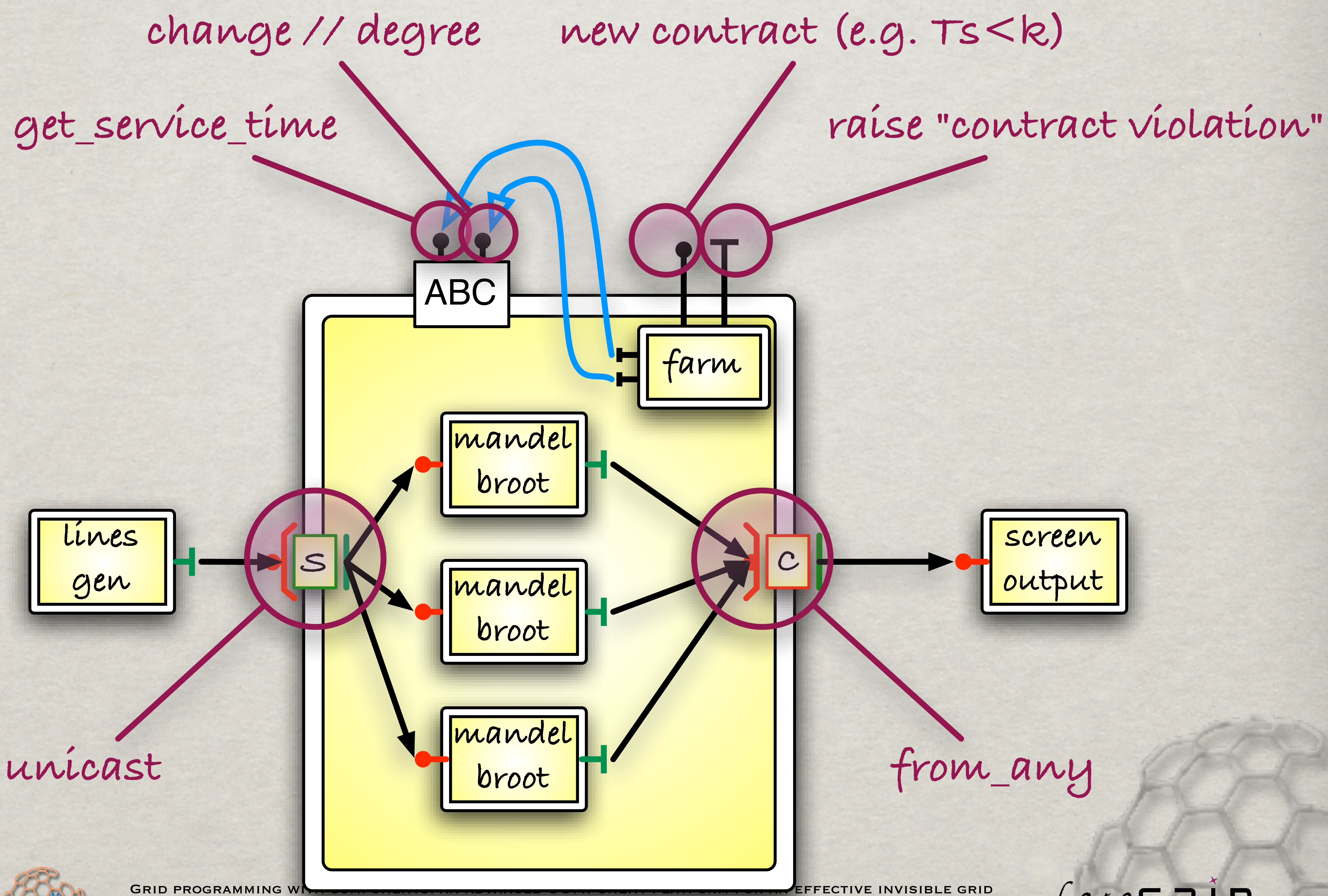
GCM IMPLEMENTATION

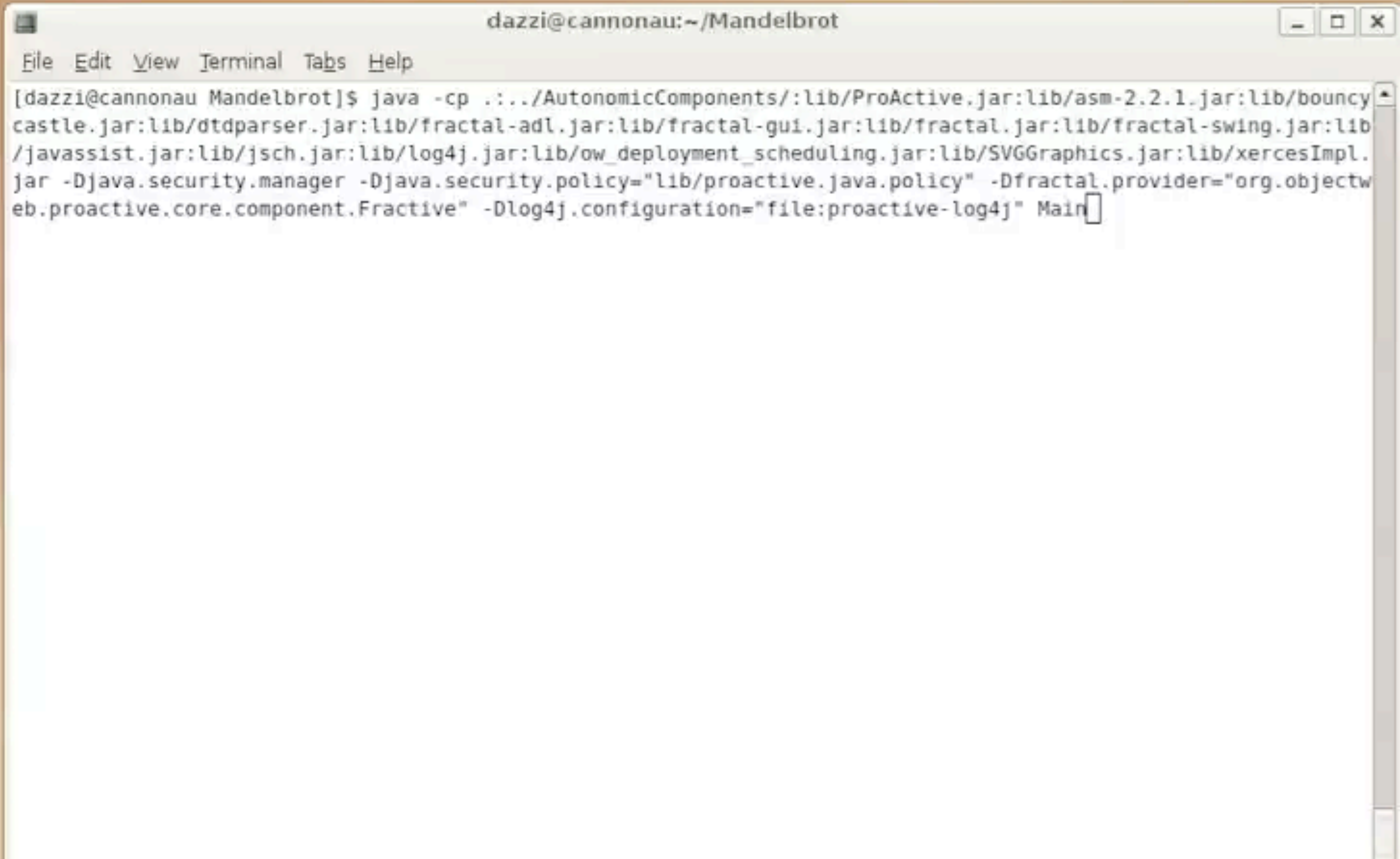


1. Choose a schema (.e.g. functional replication) ABC API is chosen accordingly
2. Choose an inner component (compliant to Be-Ske constraints)
3. Choose behavior of ports (e.g. unicast/from_any, scatter/gather)
4. Wire it in your application. Run it, then trigger adaptations
5. Possibly, automatize the process with a Manager

ABC = Autonomic Behaviour Controller (implements mechanisms)
AM = Autonomic Manager (implements policies)
B/LC = Binding + Lifecycle Controller

Farm example (Mandelbroot)



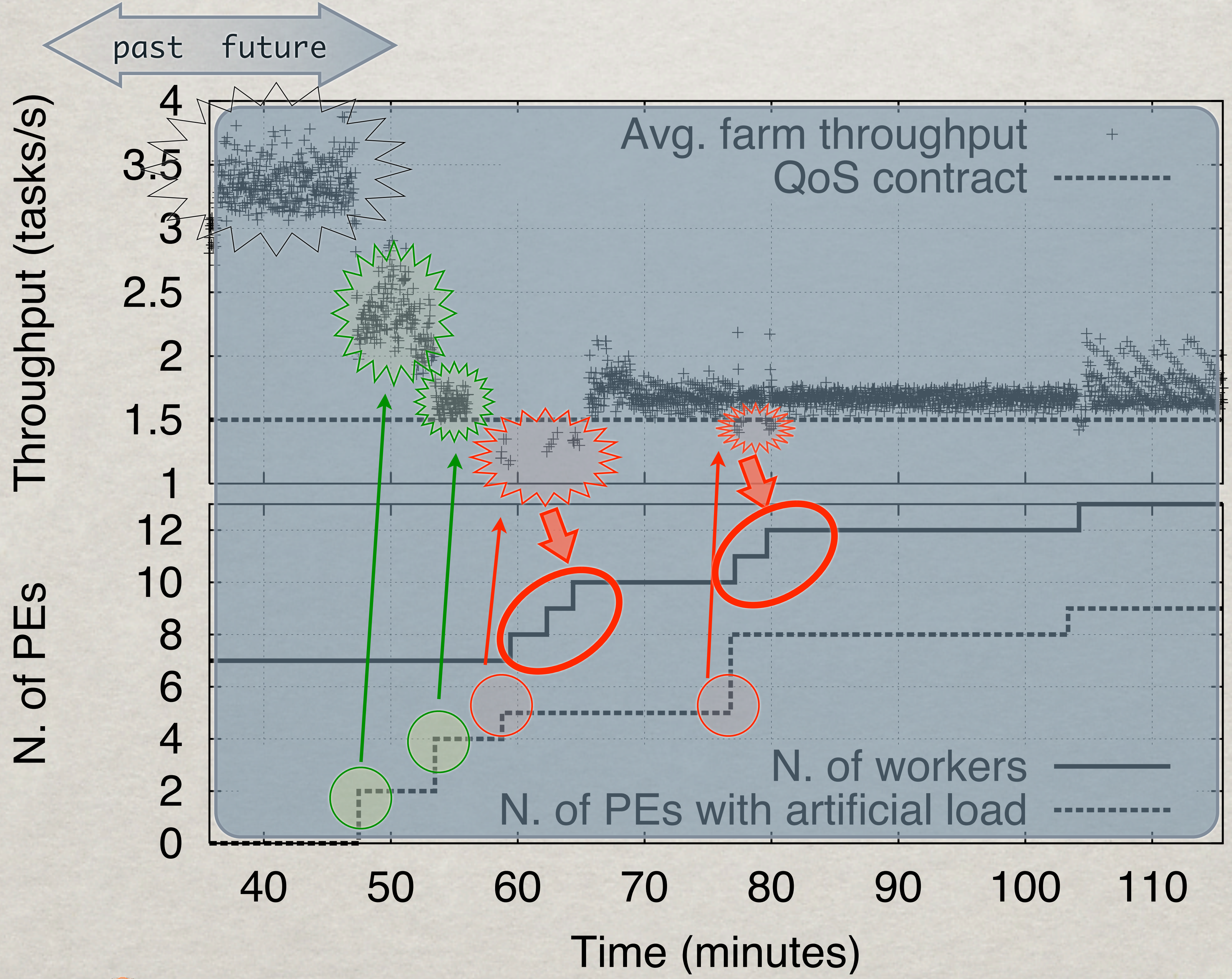


```
[dazzi@cannonau Mandelbrot]$ java -cp .:../AutonomicComponents/:lib/ProActive.jar:lib/asm-2.2.1.jar:lib/bouncy
castle.jar:lib/dtdparser.jar:lib/fractal-adl.jar:lib/fractal-gui.jar:lib/fractal.jar:lib/fractal-swing.jar:lib
/javassist.jar:lib/jsch.jar:lib/log4j.jar:lib/ow_deployment_scheduling.jar:lib/SVGGraphics.jar:lib/xercesImpl.
jar -Djava.security.manager -Djava.security.policy="lib/proactive.java.policy" -Dfractal.provider="org.objectw
eb.proactive.core.component.Fractive" -Dlog4j.configuration="file:proactive-log4j" Main
```

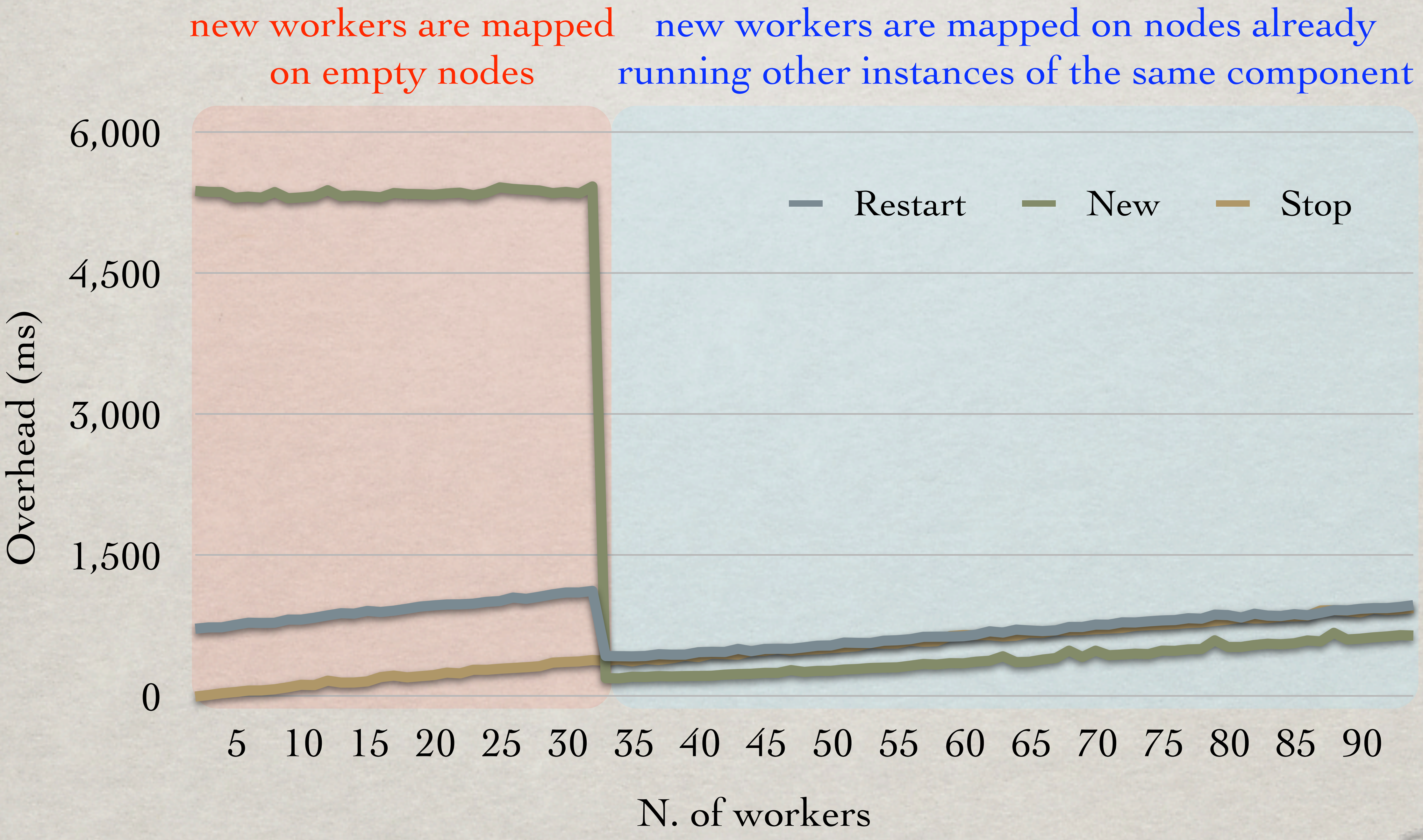
Not just farm (i.e. param sweep)

- Many other skeletons already developed for GCM
  - some mentioned before
- Easy extendible to stateful variants
  - imposing inner component expose NF ports for state access
- Policies not discussed here
  - expressed with a `when-event-if-cond-then-action` list of rules
  - some exist, work ongoing ...

Log of the Run (Explained)



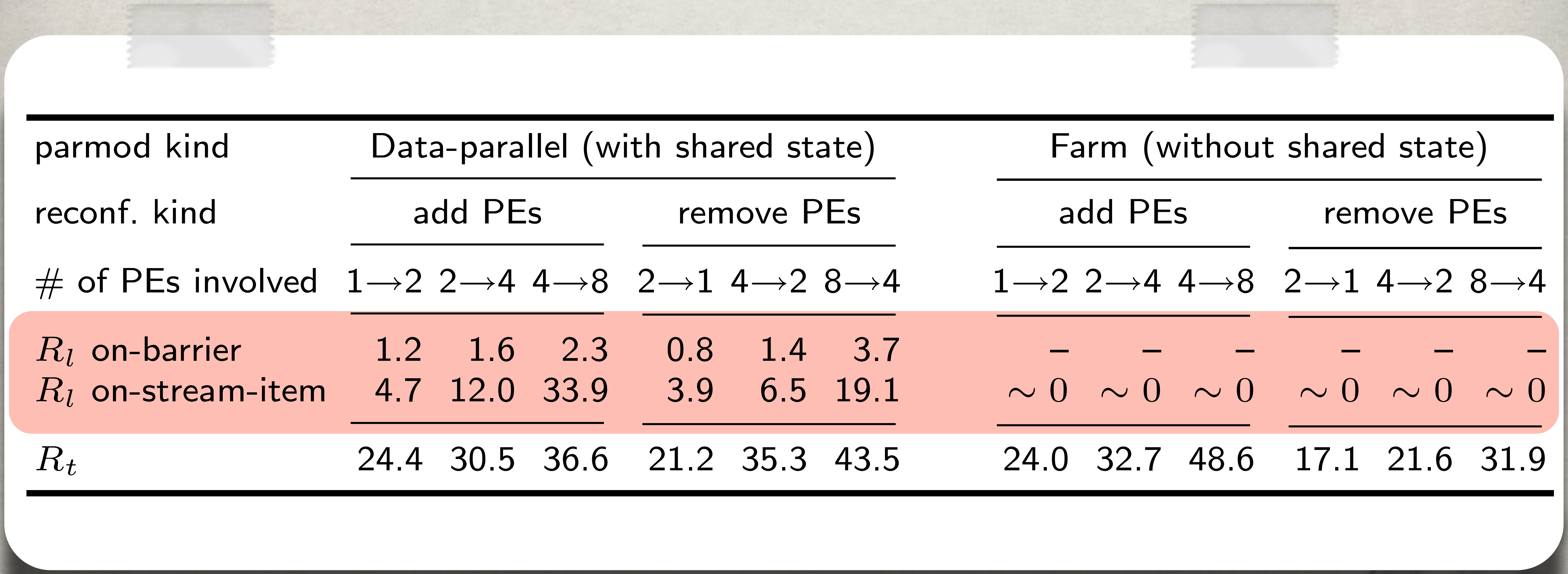
Overheads


new workers are mapped on empty nodes
new workers are mapped on nodes already running other instances of the same component
Restart
New
Stop
Overhead (ms)
6,000
4,500
3,000
1,500
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
N. of workers

ProActive/Java appears quite heavyweight w.r.t. other approaches

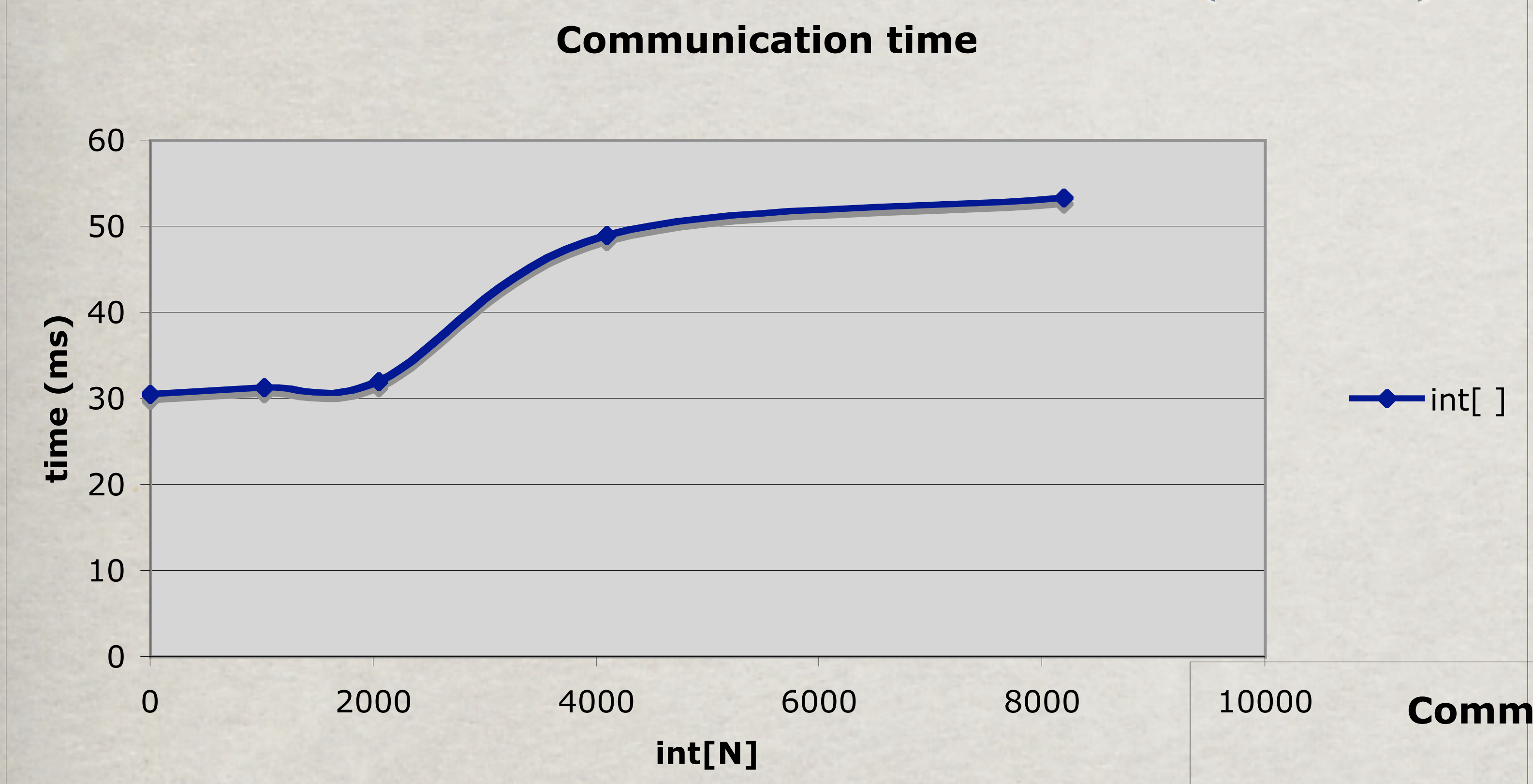
ASSIST/C++ overheads (ms)

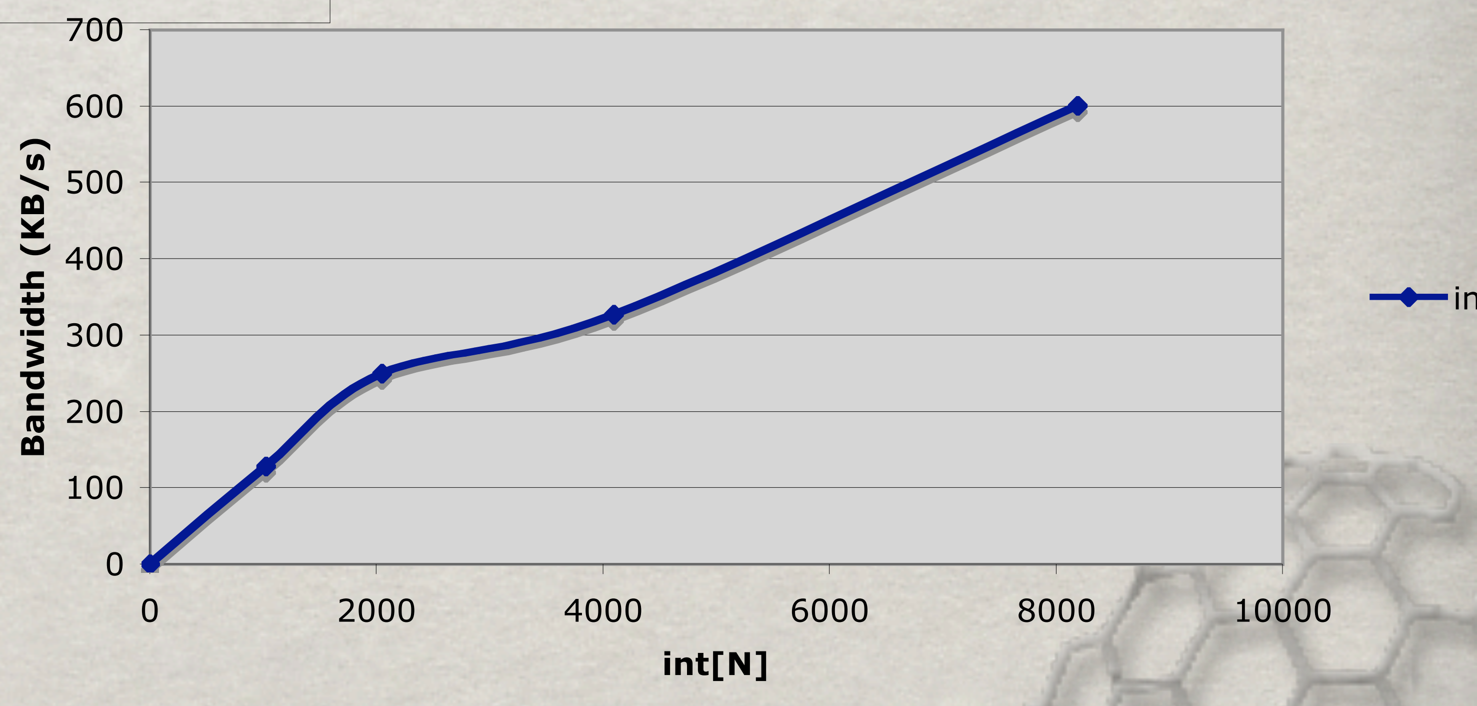
M. Aldinucci, A. Petrocelli, E. Pistoletti, M. Torquati, M. Vanneschi, L. Veraldi, and C. Zoccolo.
Dynamic reconfiguration of grid-aware applications in ASSIST.
Euro-Par 2005, vol. 3648 of LNCS, Lisboa, Portugal. Springer Verlag, August 2005.



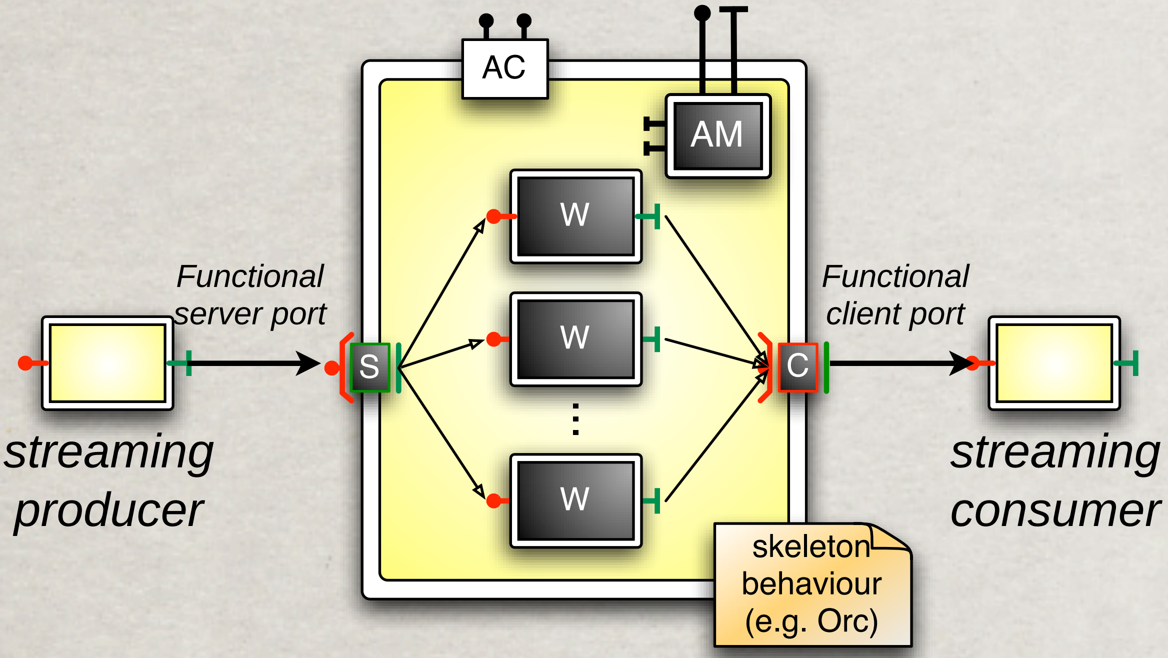
parmod kind	Data-parallel (with shared state)						Farm (without shared state)					
reconf. kind	add PEs			remove PEs			add PEs			remove PEs		
# of PEs involved	1→2	2→4	4→8	2→1	4→2	8→4	1→2	2→4	4→8	2→1	4→2	8→4
R_l on-barrier	1.2	1.6	2.3	0.8	1.4	3.7	–	–	–	–	–	–
R_l on-stream-item	4.7	12.0	33.9	3.9	6.5	19.1	~ 0	~ 0	~ 0	~ 0	~ 0	~ 0
R_t	24.4	30.5	36.6	21.2	35.3	43.5	24.0	32.7	48.6	17.1	21.6	31.9

Proactive Communication Time (Int)


Communication time


Communication Bandwidth (Theoretical 12800 KB/s)

Variations and Flavours



Functional server port

Functional client port

AC

AM

W

W

W

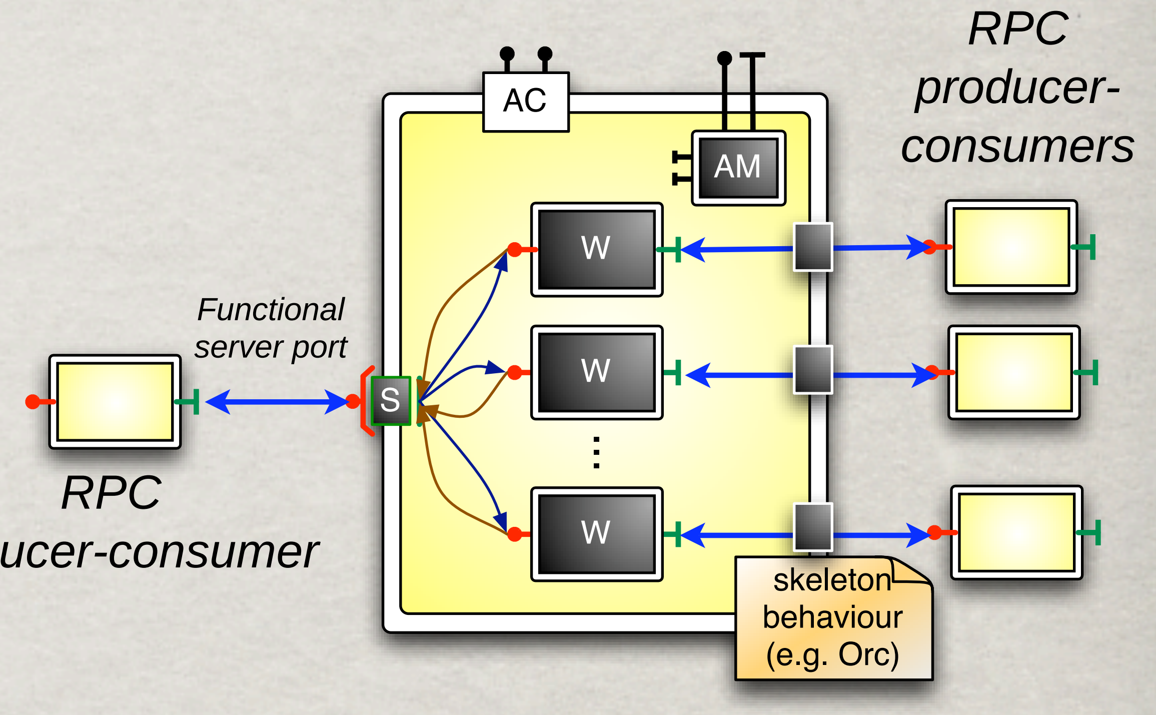
S

C

skeleton behaviour (e.g. Orc)

streaming producer

streaming consumer



RPC producer-consumers

Functional server port

AC

AM

W

W

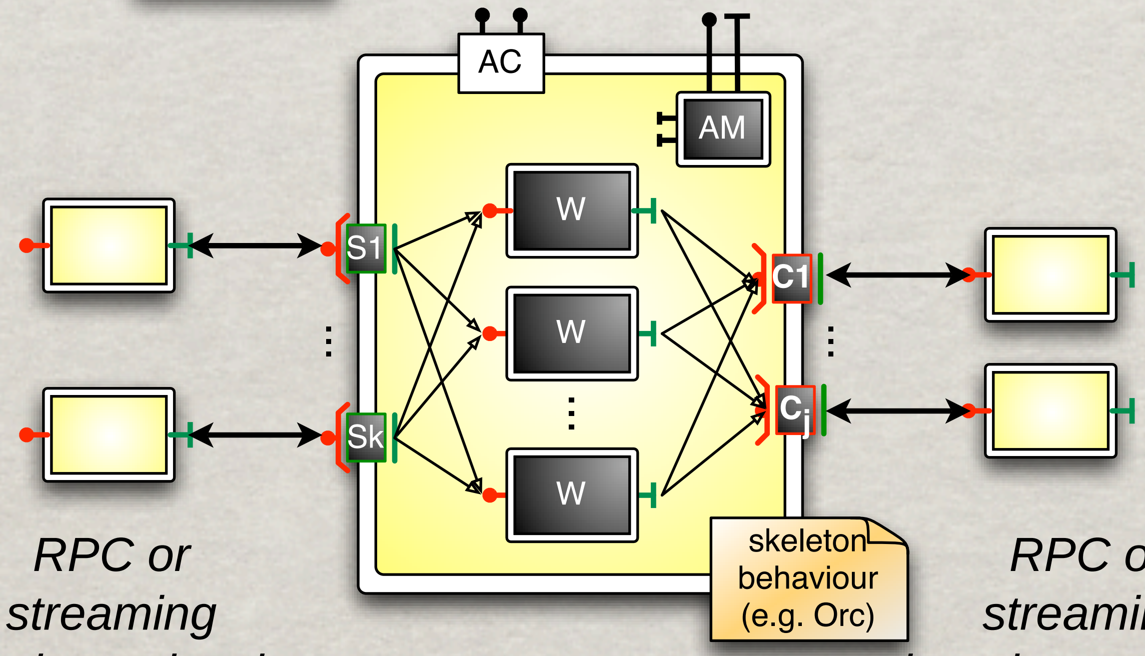
W

S

skeleton behaviour (e.g. Orc)

RPC producer-consumer

or in general ...



AC

AM

W

W

W

S1

Sk

C1

C

skeleton behaviour (e.g. Orc)

RPC or streaming data dependencies

RPC or streaming data dependencies

and even more ...

Abstracting Out Variants

- *n* client and *y* server ports
  - synchronous and/or asynchronous
  - stream and/or RPC
  - programmable, possibly nondeterministic, relations among ports
    - wait for an item on port_A ***and/or*** one item on port_B
    - in general, any CSP expression
- But ... be careful, this is the **ASSIST** model
  - all features described above + distributed membrane + autonomicity, QoS contracts, limited hierarchy depth (i.e. 2)
  - sophisticated C++ implementation, language not easy to modify
- GCM should be *enough* expressive and *not too* complex
  - we consider ASSIST as the complexity asymptote

ASSIST References (2002-07 UNIPI+ISTI/CNR only)

1. M. Aldinucci, F. André, J. Buisson, S. Campa, M. Coppola, M. Danelutto, and C. Zoccolo. Parallel program/component adaptivity management. In G. R. Joubert, W. E. Nagel, F. J. Peters, O. Plata, P. Tirado, and E. Zapata, editors, Parallel Computing: Current & Future Issues of High-End Computing (Proc. of PARCO 2005, Malaga, Spain), volume 33 of NIC, pages 89–96, Germany, Dec. 2005. John von Neumann Institute for Computing.
2. M. Aldinucci, F. André, J. Buisson, S. Campa, M. Coppola, M. Danelutto, and C. Zoccolo. An abstract schema modeling adaptivity management. In S. Gorlatch and M. Danelutto, editors, Integrated Research in Grid Computing, CoreGRID. Springer, Dec. 2006.
3. M. Aldinucci, G. Antoniu, M. Danelutto, and M. Jan. Fault-tolerant data sharing for high-level grid programming: A hierarchical storage architecture. In M. Bubak, S. Gorlatch, and T. Priol, editors, Achievements in European Research on Grid Systems, CoreGRID Series, pages 67–81. Springer, Nov. 2007.
4. M. Aldinucci and A. Benoit. Automatic mapping of ASSIST applications using process algebra. In Proc. of HLPP2005: Intl. Workshop on High-Level Parallel Programming. Warwick University, Coventry, UK, July 2005.
5. M. Aldinucci and A. Benoit. Towards the automatic mapping of ASSIST applications for the grid. In S. Gorlatch and M. Danelutto, editors, Proc. of the Integrated Research in Grid Computing Workshop, volume TR-05-22, pages 59–68, Pisa, Italy, Nov. 2005. Università di Pisa, Dipartimento di Informatica.
6. M. Aldinucci and A. Benoit. Towards the automatic mapping of ASSIST applications for the grid. In S. Gorlatch and M. Danelutto, editors, Integrated Research in Grid Computing, CoreGRID, pages 73–87. Springer, Dec. 2006.
7. M. Aldinucci and A. Benoit. Automatic mapping of ASSIST applications using process algebra. Parallel Processing Letters, 2008.
8. M. Aldinucci, C. Bertolli, S. Campa, M. Coppola, M. Vanneschi, L. Veraldi, and C. Zoccolo. Self-configuring and self-optimising grid components in the gcm model and their ASSIST implementation. Technical Report TR-06-13, Università di Pisa, Dipartimento di Informatica, Italy, Aug. 2006.
9. M. Aldinucci, C. Bertolli, S. Campa, M. Coppola, M. Vanneschi, L. Veraldi, and C. Zoccolo. Self-configuring and self-optimizing grid components in the GCM model and their ASSIST implementation. In Proc. of. HPC-GECO/Compframe (held in conjunction with HPDC-15), IEEE, pages 45–52, Paris, France, June 2006.
10. M. Aldinucci, S. Campa, P. Ciullo, M. Coppola, M. Danelutto, P. Pesciullesi, R. Ravazzolo, M. Torquati, M. Vanneschi, and C. Zoccolo. ASSIST demo: a high level, high performance, portable, structured parallel programming environment at work. In H. Kosch, L. Böszörményi, and H. Hellwagner, editors, Proc. of 9th Intl. Euro-Par 2003 Parallel Processing, volume 2790 of LNCS, pages 1295–1300, Klagenfurt, Austria, Aug. 2003. Springer.
11. M. Aldinucci, S. Campa, P. Ciullo, M. Coppola, M. Danelutto, P. Pesciullesi, R. Ravazzolo, M. Torquati, M. Vanneschi, and C. Zoccolo. A framework for experimenting with structure parallel programming environment design. In G. R. Joubert, W. E. Nagel, F. J. Peters, and W. V. Walter, editors, Parallel Computing: Software Technology, Algorithms, Architectures and Applications (Proc. of PARCO 2003, Dresden, Germany), volume 13 of Advances in Parallel Computing, pages 617–624, Germany, 2004. Elsevier.
12. M. Aldinucci, S. Campa, P. Ciullo, M. Coppola, S. Magini, P. Pesciullesi, L. Potiti, R. Ravazzolo, M. Torquati, M. Vanneschi, and C. Zoccolo. The implementation of ASSIST, an environment for parallel and distributed programming. In H. Kosch, L. Böszörményi, and H. Hellwagner, editors, Proc. of 9th Intl Euro-Par 2003 Parallel Processing, volume 2790 of LNCS, pages 712–721, Klagenfurt, Austria, Aug. 2003. Springer.
13. M. Aldinucci, S. Campa, M. Coppola, M. Danelutto, D. Laforenza, D. Puppin, L. Scarponi, M. Vanneschi, and C. Zoccolo. Components for high performance grid programming in grid.it. In V. Getov and T. Kielmann, editors, Proc. of the Intl. Workshop on Component Models and Systems for Grid Applications, CoreGRID series, pages 19–38, Saint-Malo, France, Jan. 2005. Springer.
14. M. Aldinucci, S. Campa, M. Coppola, S. Magini, P. Pesciullesi, L. Potiti, R. Ravazzolo, M. Torquati, and C. Zoccolo. Targeting heterogeneous architectures in ASSIST: Experimental results. In M. Danelutto, M. Vanneschi, and D. Laforenza, editors, Proc. of 10th Intl. Euro-Par 2004 Parallel Processing, volume 3149 of LNCS, pages 638–643. Springer, Aug. 2004.
15. M. Aldinucci, M. Coppola, S. Campa, M. Danelutto, M. Vanneschi, and C. Zoccolo. Structured implementation of component based grid programming environments. In V. Getov, D. Laforenza, and A. Reinefeld, editors, Future Generation Grids, CoreGRID series, pages 217–239. Springer, Nov. 2005.
16. M. Aldinucci, M. Coppola, M. Danelutto, N. Tonellotto, M. Vanneschi, and C. Zoccolo. High level grid programming with ASSIST. Computational Methods in Science and Technology, 12(1):21–32, 2006.
17. M. Aldinucci, M. Coppola, M. Danelutto, M. Vanneschi, and C. Zoccolo. ASSIST as a research framework for high-performance grid programming environments. Technical Report TR-04-09, Università di Pisa, Dipartimento di Informatica, Italy, Feb. 2004.
18. M. Aldinucci, M. Coppola, M. Danelutto, M. Vanneschi, and C. Zoccolo. ASSIST as a research framework for high-performance grid programming environments. In J. C. Cunha and O. F. Rana, editors, Grid Computing: Software environments and Tools, chapter 10, pages 230–256. Springer, Jan. 2006.
19. M. Aldinucci and M. Danelutto. Stream parallel skeleton optimization. In Proc. of PDCS: Intl. Conference on Parallel and Distributed Computing and Systems, pages 955–962, Cambridge, Massachusetts, USA, Nov. 1999. IASTED, ACTA press.
20. M. Aldinucci and M. Danelutto. An operational semantics for skeletons. In G. R. Joubert, W. E. Nagel, F. J. Peters, and W. V. Walter, editors, Parallel Computing: Software Technology, Algorithms, Architectures and Applications (Proc. of PARCO 2003, Dresden, Germany), volume 13 of Advances in Parallel Computing, pages 63–70, Germany, 2004. Elsevier.
21. M. Aldinucci and M. Danelutto. Algorithmic skeletons meeting grids. Parallel Computing, 32(7):449–462, 2006. DOI:10.1016/j.parco.2006.04.001.
22. M. Aldinucci and M. Danelutto. Skeleton based parallel programming: functional and parallel semantic in a single shot. Computer Languages, Systems and Structures, 2006. doi: 10.1016/j.cl.2006.07.004, in press.
23. M. Aldinucci, M. Danelutto, and J. Dünnweber. Optimization techniques for implementing parallel skeletons in grid environments. In S. Gorlatch, editor, Proc. of CMPP: Intl. Workshop on Constructive Methods for Parallel Programming, pages 35–47, Stirling, Scotland, UK, July 2004. Universität Münster, Germany.
24. M. Aldinucci, M. Danelutto, A. Paternesi, R. Ravazzolo, and M. Vanneschi. Building interoperable grid-aware ASSIST applications via WebServices. In G. R. Joubert, W. E. Nagel, F. J. Peters, O. Plata, P. Tirado, and E. Zapata, editors, Parallel Computing: Current & Future Issues of High-End Computing (Proc. of PARCO 2005, Malaga, Spain), volume 33 of NIC, pages 145–152, Germany, Dec. 2005. John von Neumann Institute for Computing.
25. M. Aldinucci, M. Danelutto, and M. Vanneschi. Autonomic QoS in ASSIST grid-aware components. In B. D. Martino and S. Venticinque, editors, Proc. of Intl. Euromicro PDP 2006: Parallel Distributed and network-based Processing, pages 221–230, Montbéliard, France, Feb. 2006. IEEE.
26. M. Aldinucci, A. Petrocelli, E. Pistoletti, M. Torquati, M. Vanneschi, L. Veraldi, and C. Zoccolo. Dynamic reconfiguration of grid-aware applications in ASSIST. In J. C. Cunha and P. D. Medeiros, editors, Proc. of 11th Intl. Euro-Par 2005 Parallel Processing, volume 3648 of LNCS, pages 771–781. Springer, Aug. 2005.
27. R. Baraglia, M. Danelutto, D. Laforenza, S. Orlando, P. Palmerini, R. Perego, P. Pesciullesi, and M. Vanneschi. ASSISTConf: A grid configuration tool for the ASSIST parallel programming environment. In Proc. of Intl. Euromicro PDP: Parallel Distributed and network-based Processing, pages 193–200, Genova, Italy, Feb. 2003. IEEE.
28. P. D'Ambra, M. Danelutto, D. di Serafino, and M. Lapegna. Advanced environments for parallel and distributed applications: a view of current status. Parallel Computing, 28(12):1637–1662, 2002.
29. M. Danelutto. Dynamic run time support for skeletons. In E. H. D'Hollander, G. R. Joubert, F. J. Peters, and H. J. Sips, editors, Proc. of Intl. PARCO 99: Parallel Computing, Parallel Computing Fundamentals & Applications, pages 460–467. Imperial College Press, 1999.
30. M. Danelutto. Adaptive task farm implementation strategies. In Proc. of Intl. Euromicro PDP: Parallel Distributed and network-based Processing, pages 416–423, La Coruna, Spain, Feb. 2004. IEEE.
31. M. Danelutto. Irregularity handling via structured parallel programming. Intl. Journal of Computational Science and Engineering, 3-4, 2005.
32. M. Danelutto. QoS in parallel programming through application managers. In Proc. of Intl. Euromicro PDP: Parallel Distributed and network-based Processing, pages 282–289, Lugano, Switzerland, Feb. 2005. IEEE.
33. M. Danelutto, C. Migliore, and C. Pantaleo. An alternative implementation schema for ASSIST parmod. In Proc. of Intl. Euromicro PDP: Parallel Distributed and network-based Processing, pages 56–63, Montbéliard, France, Feb. 2006. IEEE.
34. M. Danelutto and M. Vanneschi. A RISC approach to Grid. In B. D. Martino, J. Dongarra, A. Hoisie, L. T. Yang, and H. Zima, editors, Engineering the grid, chapter 8. ASP press, Jan. 2006.
35. M. Danelutto, M. Vanneschi, C. Zoccolo, N. Tonellotto, S. Orlando, R. Baraglia, T. Fagni, D. Laforenza, and A. Paccosi. HPC application execution on grids. In V. Getov, D. Laforenza, and A. Reinefeld, editors, Future Generation Grids, CoreGRID series, pages 263–282. Springer, Nov. 2005.
36. D. Laforenza and M. Vanneschi. Grid.it - next generation grid platforms and their applications. ERCIM News, 59:60–61, Oct. 2004.
37. S. Magini, P. Pesciullesi, and C. Zoccolo. Parallel software interoperability by means of CORBA in the ASSIST programming environment. In H. Kosch, L. Böszörményi, and H. Hellwagner, editors, Proc. of 9th Intl Euro-Par 2003 Parallel Processing, volume 2790 of LNCS, pages 679–688, Klagenfurt, Austria, Aug. 2003. Springer.
38. I. Merelli, L. Milanesi, D. D'Agostino, A. Clematis, M. Vanneschi, and M. Danelutto. Using parallel isosurface extraction in superficial molecular modeling. In 1st Intl. Conference on Distributed Frameworks for Multimedia Applications (DFMA 2005), pages 288–294, Besançon, France, 2005. IEEE.
39. N. Tonellotto, D. Laforenza, M. Danelutto, M. Vanneschi, and C. Zoccolo. A performance model for stream-based computations. In P. D'Ambra and M. R. Guarracino, editors, Proc. of Intl. Euromicro PDP 2007: Parallel Distributed and network-based Processing, pages 91–96, Napoli, Italia, Feb. 2007. IEEE.
40. M. Vanneschi. Heterogeneous HPC environments. In D. J. Pritchard and J. Reeve, editors, Proc. of 4th Intl. Euro-Par '98 Parallel Processing, volume 1470 of LNCS, pages 21–34, Southampton, UK, 1998. Springer.
41. M. Vanneschi. The programming model of ASSIST, an environment for parallel and distributed portable applications. Parallel Computing, 28(12):1709–1732, Dec. 2002.
42. M. Vanneschi. ASSIST high-performance programming environment: Application experiences and grid evolution. In J. Dongarra, D. Laforenza, and S. Orlando, editors, Recent Advances in Parallel Virtual Machine and Message Passing Interface,10th European PVM/MPI Users' Group Meeting, Venice, Italy, September 29 - October 2, 2003, Proceedings, volume 2840 of LNCS, pages 24–26, Venice, Italy, 2003. Springer.

BeSke, GCM, and ORC References GridCOMP or CoreGrid related (2007-08 UNIPI+ISTI/CNR only)

1. M. Aldinucci, S. Campa, M. Danelutto, P. Dazzi, P. Kilpatrick, and N. Tonellotto. Management in distributed systems: a semi-formal approach. In Proc. of Intl. Euromicro PDP 2008: Parallel Distributed and network-based Processing, Toulouse, France, Feb. 2008. IEEE. To appear.

2. M. Aldinucci, S. Campa, M. Danelutto, P. Dazzi, P. Kilpatrick, D. Laforenza, and N. Tonellotto. Behavioural skeletons for component autonomic management on grids. In CoreGRID Workshop on Grid Programming Model, Grid and P2P Systems Architecture, Grid Systems, Tools and Environments, Heraklion, Creete, Greece, June 2007.

3. M. Aldinucci, M. Danelutto, and P. Kilpatrick. Adding metadata to orc to support reasoning about grid programming. In T. Priol and M. Vanneschi, editors, Towards Next Generation Grids (Proc. of the CoreGRID Symposium 2007), CoreGRID series, pages 205–214, Rennes, France, Sept. 2007. Springer.

4. M. Aldinucci, M. Danelutto, and P. Kilpatrick. A framework for prototyping and reasoning about grid systems. In G. R. Joubert, C. Bischof, F. Peters, T. Lippert, M. Bücker, P. Gibbon, and B. Mohr, editors, Parallel Computing: Architectures, Algorithms and Applications (Proc. of PARCO 2007, Jülich, Germany), NIC, Germany, Sept. 2007. John von Neumann Institute for Computing.

5. M. Aldinucci, M. Danelutto, and P. Kilpatrick. Management in distributed systems: a semi-fomal approach. In A.-M. Kermarrec, L. Bougé, and T. Priol, editors, Proc. of 13th Intl. Euro-Par 2007 Parallel Processing, volume 4641 of LNCS, pages 651—661, Rennes, France, Aug. 2007. Springer.

6. M. Aldinucci, M. Danelutto, and P. Kilpatrick. Orc + metadata supporting grid programming. Technical Report TR-07-10, Università di Pisa, Dipartimento di Informatica, May 2007.

7. M. Danelutto, M. Aldinucci, and P. Kilpatrick. Prototyping and reasoning about distributed systems: an orc based framework. Technical Report TR-0102, Institute on Programming Model, CoreGRID - Network of Excellence, Aug. 2007.

8. P. Kilpatrick, M. Danelutto, and M. Aldinucci. Deriving grid applications from abstract models. Technical Report TR-0085, Institute on Programming Model, CoreGRID - Network of Excellence, Apr. 2007.

9. M. Aldinucci, C. Bertolli, S. Campa, M. Coppola, M. Vanneschi, L. Veraldi, and C. Zoccolo. Self-configuring and self-optimizing grid components in the GCM model and their ASSIST implementation. In Proc of. HPC-GECO/Compframe (held in conjunction with HPDC-15), IEEE, pages 45–52, Paris, France, June 2006.

Conclusions

- Behavioural Skeletons
  - templates with built-in management for the App designer
  - methodology for the skeleton designer
    - management can be changed/refined
    - just prove your own management is correct against skeleton functional description
  - can be freely mixed with standard GCM components
    - because once instanced, they are standard
- Already implemented on GCM
  - not happy about GCM runtime performances (can be improved)
    - We also implemented in ASSIST with different performances

Thank you!
Questions?

谢谢!
问题?