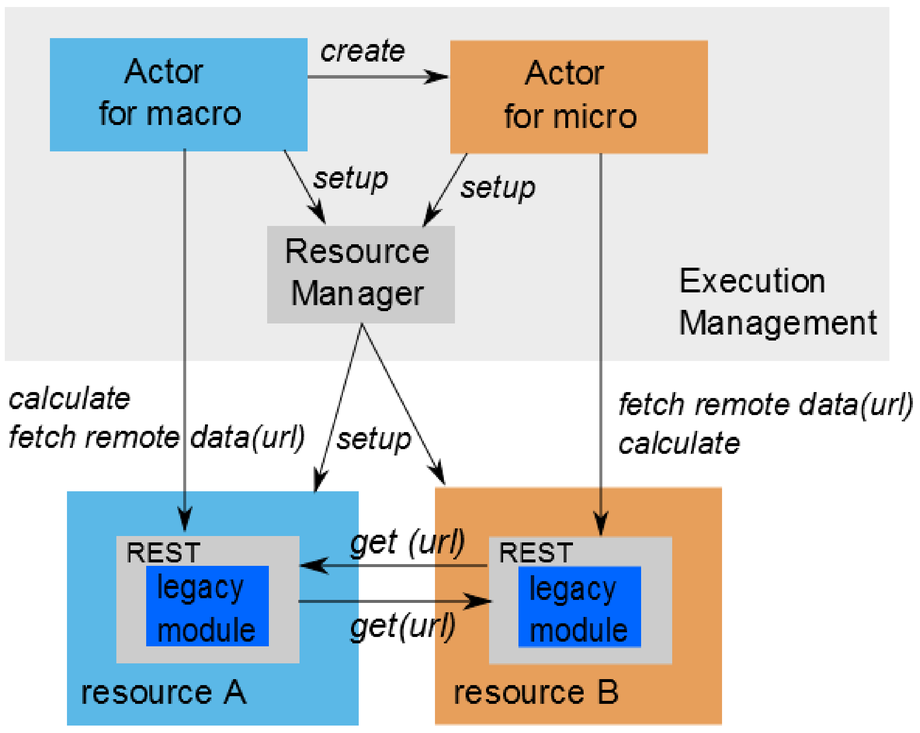
Managing Iterations Graphs in Multiscale Applications using Akka

Katarzyna Rycerz (1,2), Marian Bubak (1,2)
(1) AGH University of Science and Technology, Institute of Computer Science AGH, Department of Computer Science, al. Mickiewicza 30, 30-059 Kraków, Poland
(2) ACC Cyfronet AGH, Nawojki 11, 30-950,Kraków, Poland

Motivation

- Different coupling templates of multiscale applications [1] supported by building and execution tools [2,3].
- Efficient execution support for iteration graphs of multiscale applications on a fine-grained (iteration level)

Solution

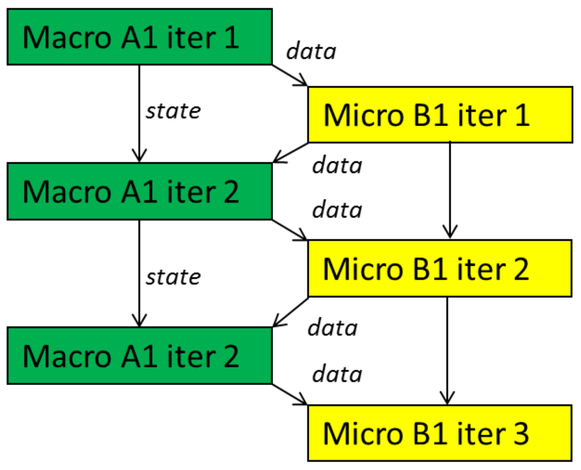


- to group modules of similar demands for the same resources
- the modules do not necessary have to belong to the same application
- the behavior of each module is modeled by Akka actors used to send commands to the wrapping service on the actual resource.
- once send to the same resources, the modules need to be properly queued to execute part of one application in the idle time of the other

Typical execution cases of macro-micro interactions

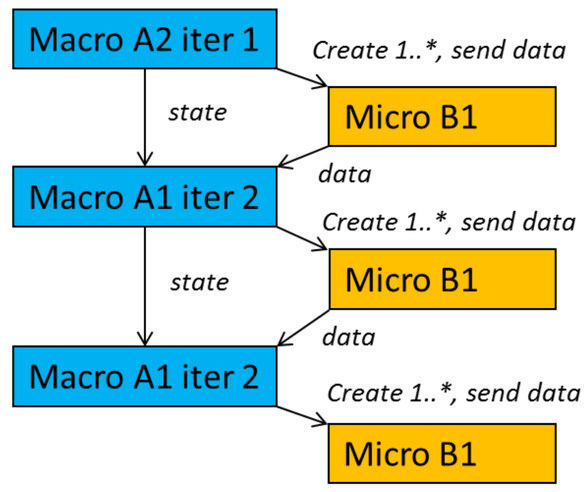
Case 1

- the macro and micro modules execute concurrently
- after each iteration the macro module sends data to the micro counterpart and waits for the results.
- The state from iteration *i* is then used in iteration *i+1*.

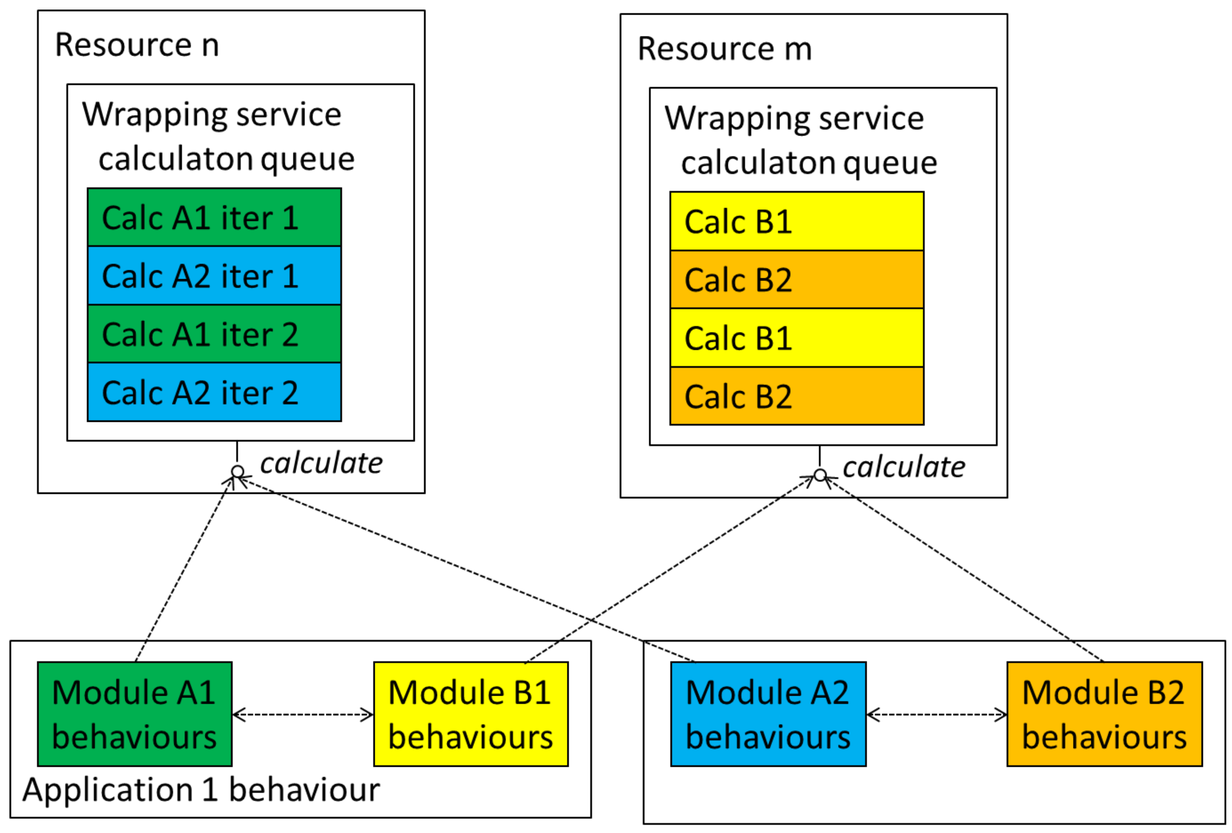


Case 2

- In each iteration macro model creates several micro modules, sends data to them and waits for the results.



- if the macro module needs to use different resources then micro module it has to wait wasting resources.
- we propose a solution that allows to efficiently use resources during these idle times.



Results

- a study with an artificial macro-micro model using Akka actors
- LU factorization algorithm with parametrized matrix size steering the CPU time used by the simulation
- comparison of execution of two macro modules on the same CPU resource with different Akka executor settings
- the results confirmed that appropriate Akka executor mechanisms allows one application to use gaps during the idle time of the other application.

	Default executor	Custom executor
Average iteration execution time	2.3 s	1.1 s
Avarage time between iteractions	2.8 s	1.3 s

Conclusions

- The preliminary results are promising for building the Akka-based framework managing execution of the applications consisting of distributed modules communicating in iterations.
- As a resources, apart from HPC we also plan to use cloud resources and mechanism of lightweight virtualization containers.
- The main challenges are to build a solution for a set of legacy multiscale applications[4].

1. Joris Borgdorff, at al: Foundations of distributed multiscale computing: Formalization, specification, and analysis, JPDC, Volume 73, Issue 4, April 2013, Pages 465-483
2. E. Ciepiela at al. Exploratory Programming in the Virtual Laboratory, in Proceedings of the International Multiconference on Computer Science and Information Technology pp. 621–628[3]
3. K. Rycerz, et al: An Environment for Programming and Execution of Multiscale Applications, Future Generation Compuetr Systems , in review
4. K. Rycerz, et al: Enabling Multiscale Fusion Simulations on Distributed Computing Resources. In: M. Bubak, J. Kitowski, K. Wiatr (Eds.): eScience on Distributed Computing Infrastructure, LNCS, Vol. 8500. Springer, pp. 195-210 (2014)

This study was partly supported by the AGH grant no 11.11.230.124 and also by the European Union within the European Regional Development Fund program no. POIG.02.03.00-12-137/13 as part of the PLGrid Core project

INNOVATIVE ECONOMY
NATIONAL COHESION STRATEGY

The work was co-funded by the European Regional Development Fund as part of the Innovative Economy program.

http://dice.cyfronet.pl

EUROPEAN UNION
EUROPEAN REGIONAL DEVELOPMENT FUND