Self-Configuration and Self-Optimization
Algorithmic Skeletons using Events

Gustavo Pabón
NIC Chile Research Labs

Ludovic Henrio
INRIA Sophia Antipolis

Part of the SCADA Associate Team: OASIS / NIC Labs
Large-scale parallel and distributed environments allow the resolution of large-scale problems.
Large-scale parallel and distributed environments allow the resolution of large-scale problems.

IBM’s Blue Gene/Q Sequoia at the Lawrence Livermore National Lab, first million core supercomputer.  
Jan/2013
Large-scale parallel and distributed environments allow the resolution of large-scale problems.

However, parallel software development is hard
Large-scale parallel and distributed environments allow the resolution of large-scale problems.

However, parallel software development is hard

and currently, we are facing an increasing challenge due to the increasing number of cores available. Indeed many-core supercomputers are almost impossible to program efficiently,
Algorithmic Skeletons: Structured Management of Parallel Computation (Research monographs in parallel & distributed computing)

Cole, Murray

Note: This is not the actual book cover
Algorithmic Skeletons: Structured Management of Parallel Computation (Research monographs in parallel & distributed computing)
Large-scale parallel and distributed environments allow the resolution of large-scale problems.

However, parallel software development is hard and currently, we are facing an increasing challenge due to the increasing number of cores available. Indeed many-core supercomputers are almost impossible to program efficiently, and those architectures are even more difficult to maintain. According to Gartner '12, IT operations management costs are the 36% of the total operation IT budget.
IBM 2001

PMAM 2014. Self-Configuration and Self-Optimization Autonomic Skeletons using Events
Self-Configuration and Self-Optimization Algorithmic Skeletons using Events
Self-Configuration and Self-Optimization

Algorithmic Skeletons using Events

PROBLEM

HIGH-LEVEL PARALLEL PROGRAMMING

MAP

Split

SEQ

Merge

Execute
Self-Configuration and Self-Optimization Algorithmic Skeletons using Events
Self-Configuration and Self-Optimization Algorithmic Skeletons using Events
Self-Configuration and Self-Optimization Algorithmic Skeletons using Events
Self-Configuration and Self-Optimization Algorithmic Skeletons using Events

Goals:
- Monitoring the Skeleton's execution.
- Creating a clear separation of concerns (SoC)

Challenge: Inversion of Control.
Self-Configuration and Self-Optimization Algorithmic Skeletons using Events

```java
class MyListener implements ... {
    public P handler(P param,
                     Skeleton[] strace) {
        ...
        return param;
    }
}

myMap.addBeforeMergeListener(myListener);
```
Let's say that

QoS Wall Clock Time (WCT): 12 seg.

WCT using 2 threads: 14 secs.
Let's say that

QoS Wall Clock Time (WCT): 12 seg.

WCT using 2 threads: 14 secs.
WCT using 4 threads: 10 secs.
Self-Configuration and Self-Optimization Algorithmic Skeletons using Events

WCT using 2 threads: 14 secs.
WCT using 4 threads: 10 secs.

How to calculate them?
Self-Configuration and Self-Optimization
Algorithmic Skeletons using Events

WCT using 2 threads: **14 secs.**
WCT using 4 threads: **10 secs.**

Activity Dependency Graph

HIGH-LEVEL PARALLEL PROGRAMMING

ACTUAL EXECUTION
Let's assume that we know in advanced the following values:

- $|fs| = 4$
- $t(fs) = 3$
- $t(fe) = 4$
- $t(fm) = 3$

WCT using 2 threads: 14 secs.
WCT using 4 threads: 10 secs.
Let's assume that we know in advanced the following values:

- $|fs| = 4$
- $t(fs) = 3$
- $t(fe) = 4$
- $t(fm) = 3$

Activity Dependency Graph

WCT using 2 threads: **14 secs.**
WCT using 4 threads: **10 secs.**
Let's assume that we know in advanced the following values:

- \( |fs| = 4 \)
- \( t(fs) = 3 \)
- \( t(fe) = 4 \)
- \( t(fm) = 3 \)

**Best Effort Estimation**

**Estimated Execution Time Line**

- \([0,3)\) 1
- \([3,7)\) 4
- \([7,10)\) 1
Let's assume that we know in advance the following values:

- \(|fs| = 4\)
- \(t(fs) = 3\)
- \(t(fe) = 4\)
- \(t(fm) = 3\)

**Estimated Execution Time Line**
WCT using 2 threads: **14 secs**.
WCT using 4 threads: **10 secs**.

Let's assume that we know in advance the following values:

- $|fs| = 4$
- $t(fs) = 3$
- $t(fe) = 4$
- $t(fm) = 3$

**Estimated Execution Time Line**

- $[0,3)$: $1$
- $[3,7)$: $2$
- $[7,11)$: $2$
- $[11,14)$: $1$
Let's assume that we know in advanced the following values:

\[ |fs| = 4 \]
\[ t(fs) = 3 \]
\[ t(fe) = 4 \]
\[ t(fm) = 3 \]

How to estimate this values on the fly?
Let's assume that we know in advanced the following values:

\[ |fs| = 4 \]
\[ t(fs) = 3 \]
\[ t(fe) = 4 \]
\[ t(fm) = 3 \]

How to estimate this values on the fly?

\[
\text{newEstimatedVal} = \rho \times \text{lastActualVal} + (1 - \rho) \times \text{previousEstimatedVal}
\]

\[ 0 \leq \rho \leq 1 \]
The executive summary of our proposal is to extend the Algorithmic Skeleton model by:

1. Introducing separation of concerns using events (monitoring)
2. Implementing autonomic concerns by using
   a. Activity dependency graphs
   b. Estimating future work based on:

\[
\text{newEstimatedValue} = \rho \times \text{lastActualValue} + (1 - \rho) \times \text{previousEstimatedValue}
\]
Aims to produce a new design and implementation process based on adaptable parallel patterns for component based architectures, where the autonomic features are more related to structural adaptations. Our approach contributes to provide autonomic features for the computational aspects, and it is not related to a specific architecture.
Related Works

ASPARA Project

H. Gonzales-Velez, M.Cole. 2010

This work proposes a methodology to introduce adaptability in skeletons. On ASPARA, structural information is introduced during compilation. Compared to ASPARA project, our solution allows the introduction of structural information during execution.
Related Works

Auto-tuning SkePU
U. Dastgeer, J. Enmyren, C.W. Kessler. 2011

Here the prediction mechanism at execution time uses online pre-calculated estimates to construct an execution plan for any desired configuration with minimal overhead. Our solution does not need pre-calculated estimates. It calculates estimates at runtime.
Problem: Calculate the top 5 of hashtags and commented users on 1.2 million of Colombian Twits from July 25/13 to August 5/13.

Used architecture:
- Intel(R) Xenon(R) E5645 a 2.4 GHz per core, 12 cores y 24 CPU Threads. 64 GB RAM
- Skandium v1.1b1 on JRE 1.6

Execution scenarios
- A achievable goal but stressful:
  - Goal of 9.5 secs without initialization
  - Goal of 9.5 secs with initialization
- A goal with clearance (not that stressful): goal of 10.5 secs
Execution Example - **Goal of 9.5 secs without initialization**

![Graph showing number of active threads over wall clock time with WCT=9.3s]
Execution Example - **Goal of 9.5 secs with initialization**

WCT=8.4s
Execution Example - **Goal of 10.5 secs (clearance)**

![Graph showing number of active threads over wall clock time](image)

WCT = 10.6s
Conclusions

- We have shown how skeletons together with autonomic computing present a promising solution for the autonomic adaptation of applications.

- Our approach relies on the use of events for creating a clear separation of concerns without lowering the high-level parallel programming of skeletons allowing us to precisely monitor the status of the execution, and react on the fly to make behavioural changes.

- We have shown the feasibility of our proposal by introducing self-configuration and self-optimization autonomic characteristics to skeletons using events related to the Wall Clock Time and Level of Parallelism QoSs.
Future Work

- Distributed Autonomic Skeletons using this approach.
- Introduce different QoS to improve scalability, maintainability and security: self-healing and self-protecting characteristics.
- Analyses of different WCT estimation algorithms comparing its overhead costs.
- More experiments are conducted on other benchmarks.
The executive summary of our proposal is to extend the Algorithmic Skeleton model by:

1. Introducing separation of concerns using events (monitoring)
2. Implementing autonomic concerns by using
   a. Activity dependency graphs
   b. Estimating future work based on:

\[
\text{newEstimatedVal} = \rho \times \text{lastActualVal} + (1 - \rho) \times \text{previousEstimatedVal}
\]