The **Generic Autonomic Management Framework (GAMF)**

Markus Tauber
www.cs.st-andrews.ac.uk/~markus
June 2010
Agenda

• What is Autonomic Management?
• What is the GAMF & what can it do for you?
• What was the GAMF already used for?
• Hands-on & Some pointer!
Autonomic Management

A control principle derived from the autonomic nervous system which:
• Allows to control facets of a target system’s operation
• In response to a changing environment
• Without human interaction

[Kephart & Chess, The Vision of Autonomic Computing]
GAMF at its core

Allows developer to focus on system specific control components and provides generic mechanisms.

Generic operations based on the autonomic control loop:

• Monitoring (generating events)
• Maintaining monitoring data
  – Thread-safe provisioning
  – Basic filtering
• Scheduling the
  – Extraction of metrics
  – Evaluation of policies
• Managing meta-data

Management of system specific control-components
GAMF’s Architecture

GAMF package provides:
- Generic management components
- Interfaces for interaction between generic and system adapter
- Examples system adapter
Working Examples

• Controlling the interval with which routing tables in individual P2P/Key Based Routing (KBR) overlay nodes are maintained.

• Controlling the degree of concurrency (DOC) in data retrieval operations in a distributed storage system.
Autonomic Management of Maintenance Scheduling in P2P nodes

- Extracting metrics by aggregating monitored information over a management cycle

- Non-effective maintenance operations
- Peer access errors

- Sub-policies determine a change of the interval proportionally to the amount a metric diverts from an “ideal” value - per metric
- Averaging policy balances out varying requirements

- Applying new interval via an effector

Maintenance mechanism scheduler in individual P2P nodes
Autonomic Management of the DOC in Data Retrieval Mechanisms

extracting metrics by aggregating monitoring information over a management cycle:
• DOC
• Fetch Failure Rate (FFR)
• Network Speed Variation (NSV)
• Bottleneck Location

• Degree of Concurrency (DOC)
• fetch operations
• bandwidth and latency

high level policy determines a new DOC:
• if the DOC is low, and FFR and NSV are high DOC++
• if the DOC is high, and FFR and NSV are low, and a bottleneck exists on the client side DOC--

applying new DOC via an effector
Hands-on!

www-systems.cs.st-andrews.ac.uk/gamf

• Documentation
• API
• Examples
• Downloads
Pointers

- markus@cs.st-andrews.ac.uk
- www-systems.cs.st-andrews.ac.uk/gamf (sf)
- www.cs.st-andrews.ac.uk/~markus
Maintenance in Chord

- The peer-set may not be completely accurate and result in failure and inefficient lookups
- Maintenance operations in Chord nodes periodically refresh the peer-set
- The shorter the interval between maintenance operations the more accurate the peer-set and the more resources are used
- Existing P2P overlays use statically set maintenance intervals
Problem Definition

Optimal maintenance intervals depend on the workload and churn in network membership:

- Performance and resource consumption suffers as a low maintenance interval uses resources unnecessarily under low churn or no workload.

- Performance suffers as a high maintenance interval may cause increased routing cost due lookup repetitions and errors which result from invalid peer-sets under high churn.

These unsatisfactory situations were corrected by autonomic management of the maintenance interval.
Degree of Concurrency in Data Retrieval Mechanisms

In a distributed storage system every client can fetch up to \( R \) replicated data items.

The Degree of Concurrency (\textbf{DOC}) defines the eagerness with which a client fetches data items.

A high DOC may compensate for fetch failures and variability but also may jeopardise performance as simultaneous fetch operations compete for network resources.
Problem Definition

Time to get a data item depends on:
• Degree Of Concurrency (DOC)
• Bandwidths & Latencies
• Fetch Failure Rate
• Knowledge

The ideal DOC depends on various conditions and cannot be predicted if they vary

• Performance suffers if the DOC is low and there is a large variation in times taken to fetch a replicated data item or a high failure rate

• Performance and resource consumption suffers if no variation or failures are observed and the DOC is high. Here it represents pure overhead and parallel fetch operations have to compete for network resources, especially if a network bottleneck exists near to the client

Autonomic management of the DOC was able to correct those unsatisfactory situations