JÜLICH FORSCHUNGSZENTRUM

# A highly configurable and efficient simulator for job schedulers on supercomputers

April 12, 2013 | Carsten Karbach, Jülich Supercomputing Centre (JSC)

# Motivation

## Objective

- **Simulation** of supercomputer job schedulers for prediction of job start times
- **Features** of simulation program:
  - Efficient and configurable
  - Extensible and generic
- **Use cases**:
  - User → predicts start time of his own jobs
  - Administrator → configurable simulation of supercomputer, throughput optimization
- **Problem**: unpublished scheduling algorithms, job schedulers do not provide global on-line prediction

# Motivation

## Solution

- **JuFo** (Jülich Forecast):
  C++ application using data format **LML**
- Based on prediction component of **LLview**
- LLview provides status information of supercomputers in LML
- Abstraction of scheduling systems Moab and Loadleveler

Mitglied der Helmholtz-Gemeinschaft

# Part I: Problem definition

April 12, 2013 | Carsten Karbach, Jülich Supercomputing Centre (JSC)

# Problem definition

## What is the objective of JuFo?

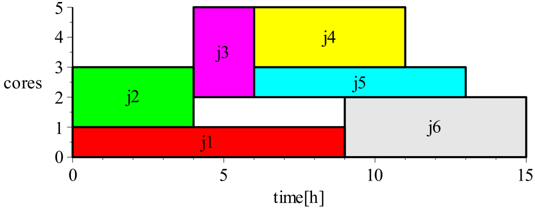
- **On-line simulation** of global job schedulers
- **LML** as interface to LLview, input and output of JuFo
- Prediction of job dispatch time and used resources

## global job scheduler

- Prioritizes and places jobs
- Fixed CPU set is allocated to each job
- Handling of reservations, queues and dependencies
- Examples: Moab (JUROPA), Loadleveler (JUQUEEN)

# The scheduling problem

- Supercomputer **offers resources**
  e.g. CPUs, GPUs, memory
- Jobs **request resources**, wait in queues
  until resources become available
- Problem: generate optimal schedule for all waiting jobs
- Optimization over time and resources
- **Approximation** with FCFS, List-Scheduling and Backfilling

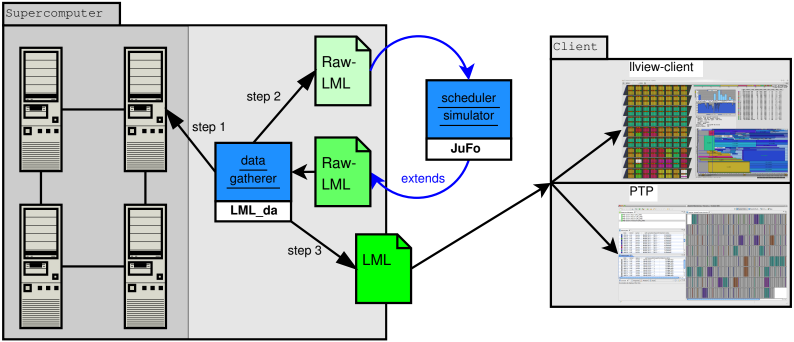


JÜLICH FORSCHUNGSZENTRUM

# Part II: Embedding JuFo

April 12, 2013 | Carsten Karbach, Jülich Supercomputing Centre (JSC)

# Embedding JuFo



JÜLICH FORSCHUNGSZENTRUM

# Data format

## Large-scale system Markup Language (LML)

- Data format for **status information of supercomputers**
- Based on XML, specified by an **XML Schema**
- Interface for LML_da, JuFo and Clients
- LLview client visualizes JuFo's outputs

## Input data required by JuFo

- Compute nodes: number of processors per node
- Queues: grouping of jobs, scheduling rules
- Jobs: running and waiting; requested resources

# Part III: Scheduling algorithms

April 12, 2013 | Carsten Karbach, Jülich Supercomputing Centre (JSC)

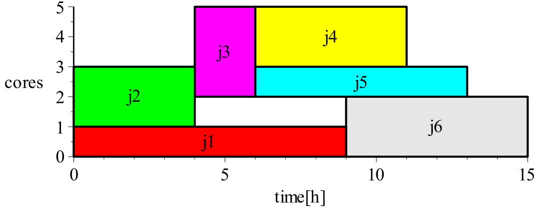
JÜLICH FORSCHUNGSZENTRUM

# Constraints

- **No job migration**
- **No preemption**
- **Wall clock limit** (WCL) is mandatory for each job
- WCL is used as actual job run-time in JuFo

# Scheduling Algorithms

- **FCFS**: Job with longest queue time starts first
- **List-Scheduling**: Prioritization by arbitrary formula, lower ranked jobs can run before higher ranked jobs
- **Backfilling**: Reservations for top dogs, fill idling resources with smaller jobs



JÜLICH FORSCHUNGSZENTRUM

# Further requirements

- Prioritization JUROPA:
  $systemprio + userprio + 10 * nodes + qtime + qtime/wall - 300 * actjobs$
- Job dependencies
- **Reservations**
- Resource requests: nodes, CPUs, GPUs, global/per node, network topology
- **Nodesharing**
- Top dogs per queue
- Queue defines scheduling constraints:
  - Allowed nodes
  - Limits number of active/waiting jobs

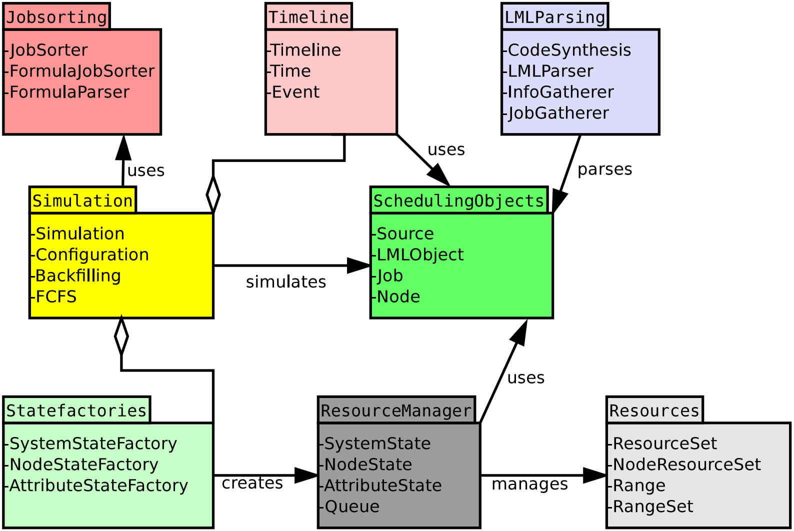
Mitglied der Helmholtz-Gemeinschaft

# Part IV: Design of JuFo

April 12, 2013 | Carsten Karbach, Jülich Supercomputing Centre (JSC)

JÜLICH
FORSCHUNGSZENTRUM

# Package overview



# Target

- Packages are **encapsulated**
- Interaction only via **interfaces**, actual implementation unknown to foreign packages
- **Reference implementation** for each interface

## Result

- Extensible **basis** for job scheduler simulation
- Well defined **extension points** for new sorting/scheduling algorithms and resource managers
- Arbitrary **combination** of available sorting/scheduling algorithms and resource managers

# Part V: Validation

April 12, 2013 | Carsten Karbach, Jülich Supercomputing Centre (JSC)

# How to validate results?

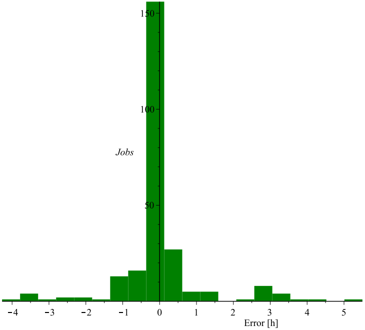
1. **Log all events** for a given time span
   (new jobs, early job completion, canceling waiting jobs)
2. Run JuFo for this time span using **exact WCLs**
3. **Compare** reality with prediction

# Validation results for JUROPA

- Validation framework
- JuFo is best configured for JUROPA
- Validation run on 8 different days

## Results

- **Time span**: 2 - 5 hours
- **Jobs**: 100-280 per day
- **Ø Error**: 1-13 minutes
- High variance due to missing information



Mitglied der Helmholtz-Gemeinschaft

JÜLICH
FORSCHUNGSZENTRUM

# Part VI: Outlook

April 12, 2013 | Carsten Karbach, Jülich Supercomputing Centre (JSC)

# Outlook

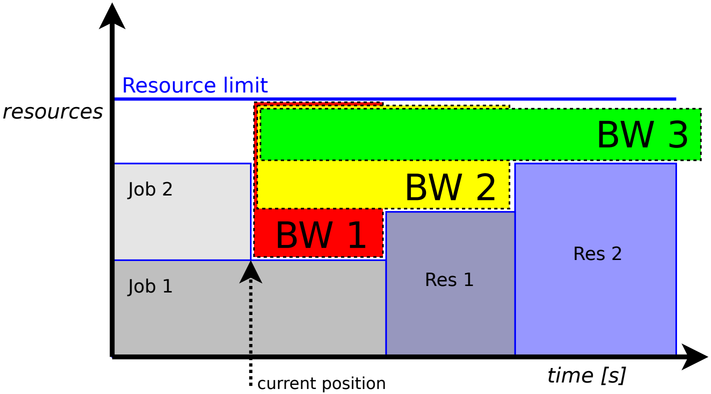
- Configuration and validation for JUQUEEN, JUDGE
- **Parallelization** for higher efficiency
- **Visualization**: in PTP, new visualization methods

Thank you for your attention!

# Summary

- **JuFo**: extensible simulation for global job schedulers
- **LML** as data format
- **LML_da** collects input data
- Scheduling algorithms:
  FCFS, List-Scheduling, Backfilling
- Analysis of Loadleveler and Moab
  $\rightarrow$ Simulation design
- Validation framework, successful on JUROPA

# Backfilling implementation



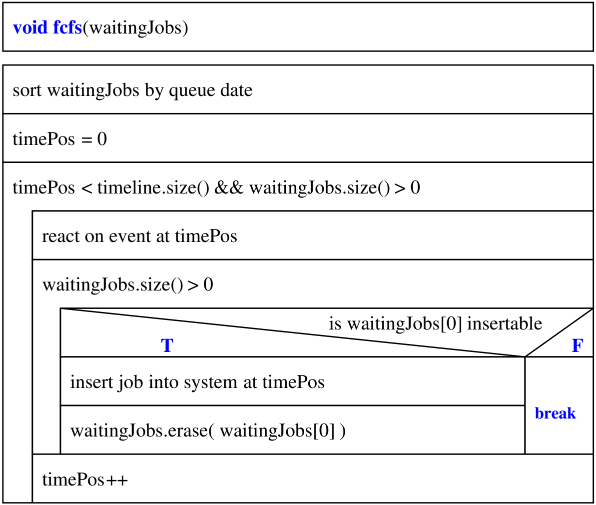

- **Backfill window**: resources × time span

JÜLICH FORSCHUNGSZENTRUM

# Scheduling problem – mathematical definition

- Schedule $s = (s_1, \ldots, s_n)^T$ with $s_j$ as dispatch time of job $j$
- $V = \{s \in \mathbb{R}^n_{\geq 0} | s \text{ is valid}\}$, all allowed schedules
- $f$ : objective function, e.g. $f(s) := -\max_{j \in J}(s_j + w_j)$, $w_j$ WCL j
- find $s_{opt}$ with $f(s_{opt}) = \max_{s \in V} f(s)$
- Derived problem from *resource-constrained project scheduling problem (RCPSP)*
- RCPSP is **NP-hard** $\rightarrow$ no solution in polynomial time
- Instead approximation with
  - First-Come-First-Served (FCFS)
  - List-Scheduling
  - Backfilling

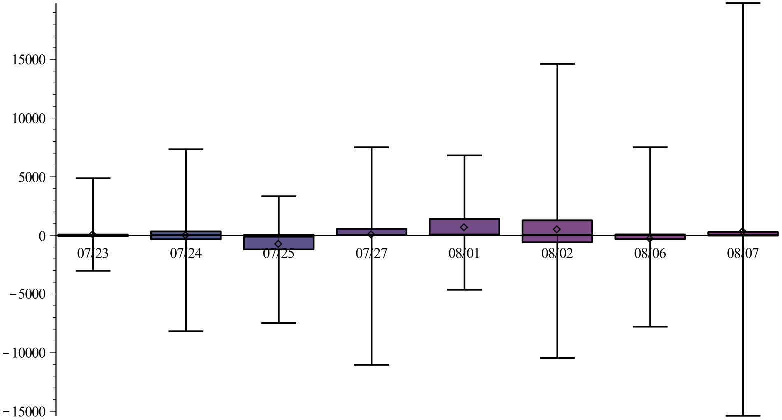
# Implementation FCFS



# Summary of JuFo packages

- **Simulation**: implements scheduling algorithms
- **ResourceManager**: manages compute resources; decides, whether a job can be started on given resources
- **Jobsorting**: configurable prioritization of waiting jobs
- **Timeline**: stores all simulation events such as job dispatch, completion and reservations
- **LMLParsing**: reads input LML, converts into object hierarchy, generates output LML

# Validation results JUROPA – Details I



## Validation results JUROPA – Details II

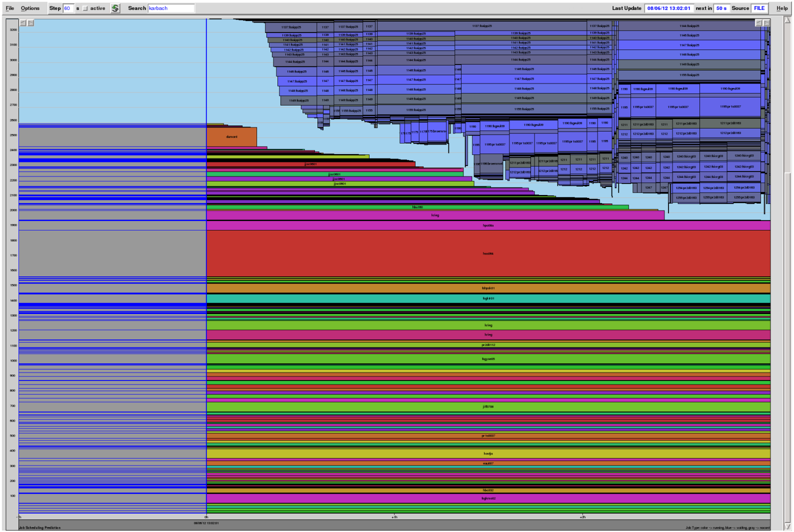
- Difference between users' WCL and actual job time span: on average 2-4 hours
- Accuracy of prediction based on users' wall comp. bad (average error of multiple hours)
- Target of JuFo is not exact prediction, but modeling the job scheduler based on input data provided by LML_da
- Better results expected by combining JuFo with **statistical data** for WCL

# JuFo for JUROPA – improvements

- Use actual contingent data for job priorities instead of *showq* outputs $\rightarrow$ system specific extension of LML_da
- Jobs with **identical system priority** are sometimes sorted wrong $\rightarrow$ improve details of job sorting
- **Reservations** are not collected by LML_da
  $\rightarrow$ extension of LML_da
- Dynamic querying of **queue-configuration**
  $\rightarrow$ extension/configuration of LML_da

# JUROPA – prediction

# JUROPA – actual schedule



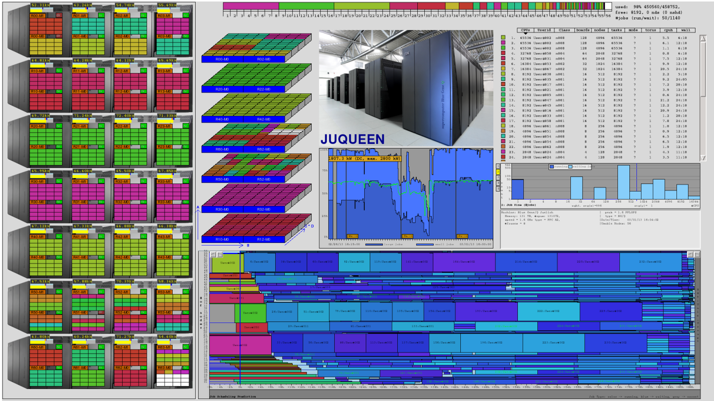
JÜLICH FORSCHUNGSZENTRUM

# Part I: LLview's prediction

April 12, 2013 | Carsten Karbach, Jülich Supercomputing Centre (JSC)

# LLview

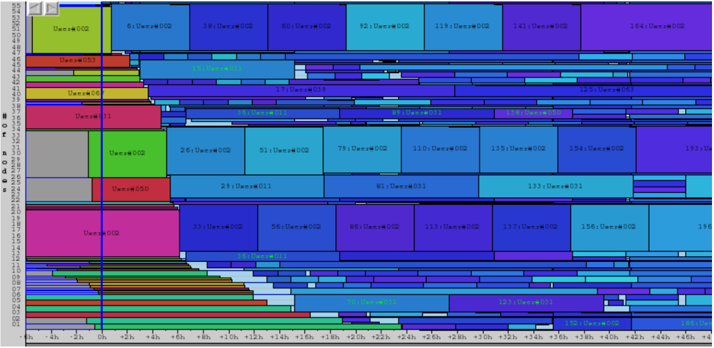
- monitors supercomputer status




Source: Snapshot LLview for JUQUEEN (IBM BG/Q, 450k cores, 5.9 Petaflops)

# LLview's prediction – SchedSim

- **SchedSim** is written in Perl
- Especially designed and tested for Loadleveler systems
- Basis for design of JuFo



Source: Snapshot LLview for JUQUEEN

# SchedSim – Analysis

## Pro

- + Highly configurable
- + Advanced modeling of JUGENE
- + Simple installation, embedded into LLview

## Contra

- – Limited scalability and performance
- – Hard to extend
- – Based on old implicit XML format,
  which is to be replaced by LML

Mitglied der Helmholtz-Gemeinschaft

JÜLICH FORSCHUNGSZENTRUM

# Part II: Job schedulers

April 12, 2013 | Carsten Karbach, Jülich Supercomputing Centre (JSC)

# Loadleveler vs. Moab

## Loadleveler

- JUGENE, JUQUEEN
- Consists of scheduler, execution machine and central manager
- Considers torus-network
- No simulation mode, prediction only for top dogs
- Backfilling

## Moab

- JUROPA, JUDGE
- Scheduler using Torque as resource manager
- Network irrelevant
- Simulation mode and *showstart*
- Backfilling with backfill window first

# Similarities

- **Reservations**
- Highly configurable
- Compute nodes offer resources, jobs request them
- Main components:
  1. Job sorting
  2. Scheduling algorithm
  3. Resource manager

Mitglied der Helmholtz-Gemeinschaft

JÜLICH FORSCHUNGSZENTRUM

# Part III: Optimization

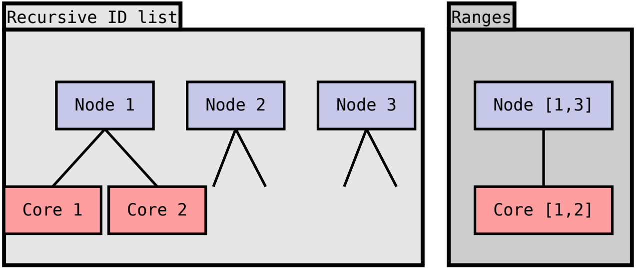
April 12, 2013 | Carsten Karbach, Jülich Supercomputing Centre (JSC)

# Serial Optimization

- **Similar jobs**: take advantage of jobs with similar requests (queue, number CPUs, WCL)
- **Simultaneous events**: execute events with identical time stamp in one iteration; reduces overall number of iterations
- **Backfill windows**: generate backfill windows and place jobs into them, instead of forward simulation separately for each job
- **Recursive interval data structure**: Use intervals for storing compute resources instead of expanded trees

# Recursive interval data structure

- JuFo manages compute nodes and CPUs per node
- Jobs are allocated to resources,
  e.g. job 1 uses nodes 10-15
- Idea 1: **Tree** → simple, but not efficient and memory demanding
- Idea 2: **Recursive intervals** → efficient, but more complex operations



# Profile analysis I

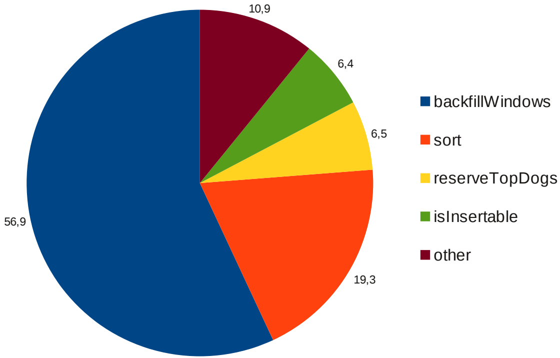
## Problem

- How much simulation time is spent for each component?

## Solution

- Profile analysis with **gprof**
- Investigate JuFo runs on JUROPA: Backfilling, 1 top dog per queue, divided into JSC and HPC-FF
- 10 input samples with 900-1600 jobs, more than 3000 nodes, simulation time 16-65 s

# Profile analysis II




## Results

- Bigger part for backfill windows
- But efficient job placement
- 20% for sorting

# Ideas for parallelization

- **I/O**: Parallel parsing of job and node data, parallel output
- **Job sorting**: job priorities are calculated independently $\rightarrow$ parallel sorting
- **Scheduling algorithms**: parallel search for suitable jobs at each time step
- **Resource manager**: parallel search for suitable compute nodes, torus search