Bachelor Thesis
Expansion of a Task Distribution Framework

TECHNISCHE
UNIVERSITÄT
DARMSTADT

by Georg Heesch
Advisor: Benjamin Schiller

Outline

TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Background
2. Goal
3. Design
4. Implementation
5. Conclusion

Outline

TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Background
2. Goal
3. Design
4. Implementation
5. Conclusion

Background - Task Distribution Systems I

TECHNISCHE UNIVERSITÄT DARMSTADT

Certain kind of Problems

- Huge tasks
  - too big for single Computer
- Parallelization possible
- relatively small subtasks

Solution: Task distribution

- multiple Computers
- Breaking up the Task into small Subtasks
- all Computers work on Subtasks
- many small computers possibly cheaper than one big one
- more reliable due to redundancy

Background - Task Distribution Systems

TECHNISCHE UNIVERSITÄT DARMSTADT

- existing Task Distribution Framework by Jan Dillmann
- Shared memory distributed Sytem
- User inserts Task information into Database
- Clients fetch task information from passive server
- executables fetched from external source
- Clients push task result to server

Problems

TECHNISCHE UNIVERSITÄT DARMSTADT

- no convenient way to interact with system
- Clients access database directly
- Tasks fetched from Database one by one
- User has to write and read Database directly
- only raw result data
- no information about eventual fails except output from Worker

Outline

TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Background
2. Goal
3. Design
4. Implementation
5. Conclusion

Goal

TECHNISCHE UNIVERSITÄT DARMSTADT

- introduce Task Lists
- Design and implement commandline user interface to system
  - must be suitable for automation
- implement Logging to collect additional data
- use Logging data to make statistical analysis
- present Results of analysis in easily accessible form

Outline

TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Background
2. Goal
3. Design
4. Implementation
5. Conclusion

Design I - Tasks

TECHNISCHE UNIVERSITÄT DARMSTADT

- Clients no loger handle single Tasks
- Lists of tasks are created when (re)queueing
- failure of a single Task has no impact on rest of List

Design II - Server Actions

TECHNISCHE UNIVERSITÄT DARMSTADT

- Creation and management of Tasks done via command line tools
- one executable for each action
- input and output via stdin/stdout
  - easily parsable and human readable format
- configuration in textfile (host, port, password, etc)

Design III - Logging

TECHNISCHE UNIVERSITÄT DARMSTADT

- As simple as possible
- a few messagetypes
- timing relevant
  - for example: duration=endTimestamp-startTimestamp

Design IV - Statistics

TECHNISCHE UNIVERSITÄT DARMSTADT

- based on Logfiles
- Modular design
- every step one module
  - calculations in one module
  - presentation in another module

Outline

TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Background
2. Goal
3. Design
4. Implementation
5. Conclusion

Implementation I - Server Actions

TECHNISCHE UNIVERSITÄT DARMSTADT

- Each Server action is an independent executable with standardized interface
- input and output of server actions in a format that is both machine readable and human readable
  - compromise found with JSON
- implemented in Java to maximize use of existing functionalities

Implementation II - Task Lists

TECHNISCHE UNIVERSITÄT DARMSTADT

- Server actions collect tasks in lists, so that a client may fetch more than one task.
- User has only rudimentary control over the structure of the Lists
  - length of lists configurable for server action
  - Sorting of tasks after single attribute (e.g. closest deadline first)
  - choice of strategy to handle lists which are not full

Implementation III - Logging

TECHNISCHE UNIVERSITÄT DARMSTADT

- extremely simple
  - a daemon listens for strings on a configurable socket
  - clients send their messages to the socket
  - daemon writes strings to file
  - logfile name is configurable and dependent on date and time
- chosen over more complicated solutions for simplicity
  - for example: syslog is more complex than our whole server structure

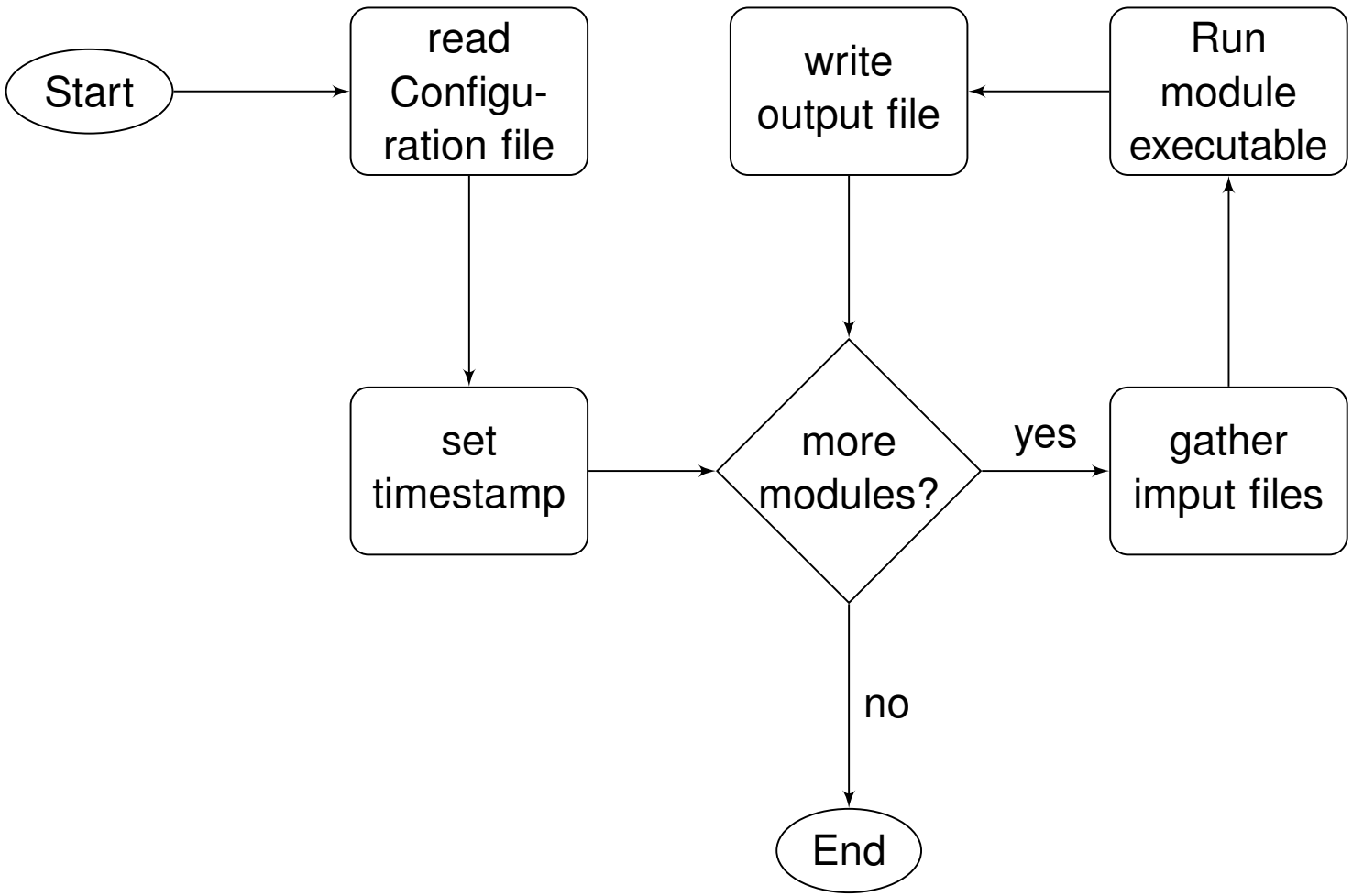
Implementation IV - Statistics

TECHNISCHE UNIVERSITÄT DARMSTADT

- a module is an executable
- a wrapper handles module execution
- modules operate on input files given via commandline
- modules produce one output file via stdout
- wrapper writes output to file

Implementation V - Wrapper

TECHNISCHE UNIVERSITÄT DARMSTADT



Implementation VI - Modules I

TECHNISCHE UNIVERSITÄT DARMSTADT

```
#!/bin/bash
tmpfile=‘mktemp‘
cat "$@" > $tmpfile

delim=","

cut -d"$delim" -f3 $tmpfile | uniq | while read host ; do
echo -n "$host "
grep $host$delim $tmpfile | grep -ic Task_Claim | tr -d "\n"
echo -n " "
grep $host$delim $tmpfile | grep -ic Task_Success | tr -d "\n"
echo -n " "
grep $host$delim $tmpfile | grep -ic Task_Fail

done | sort
rm $tmpfile
```

Implementation VII - Modules II

TECHNISCHE UNIVERSITÄT DARMSTADT

```
Host1 256 212 44
Host2 284 156 128
Host3 272 240 32
Host4 307 257 50
Host5 302 259 43
```

Outline

TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Background
2. Goal
3. Design
4. Implementation
5. Conclusion

Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

- System now has simple User Interface
- highly configurable Statistics