Software Engineering and Distributed Systems

VERKFRÆÐI- OG NÁTTÚRUVÍSINDASVIÐ www.hi.is

HÁSKÓLI ÍSLANDS

Helmut Neukirchen

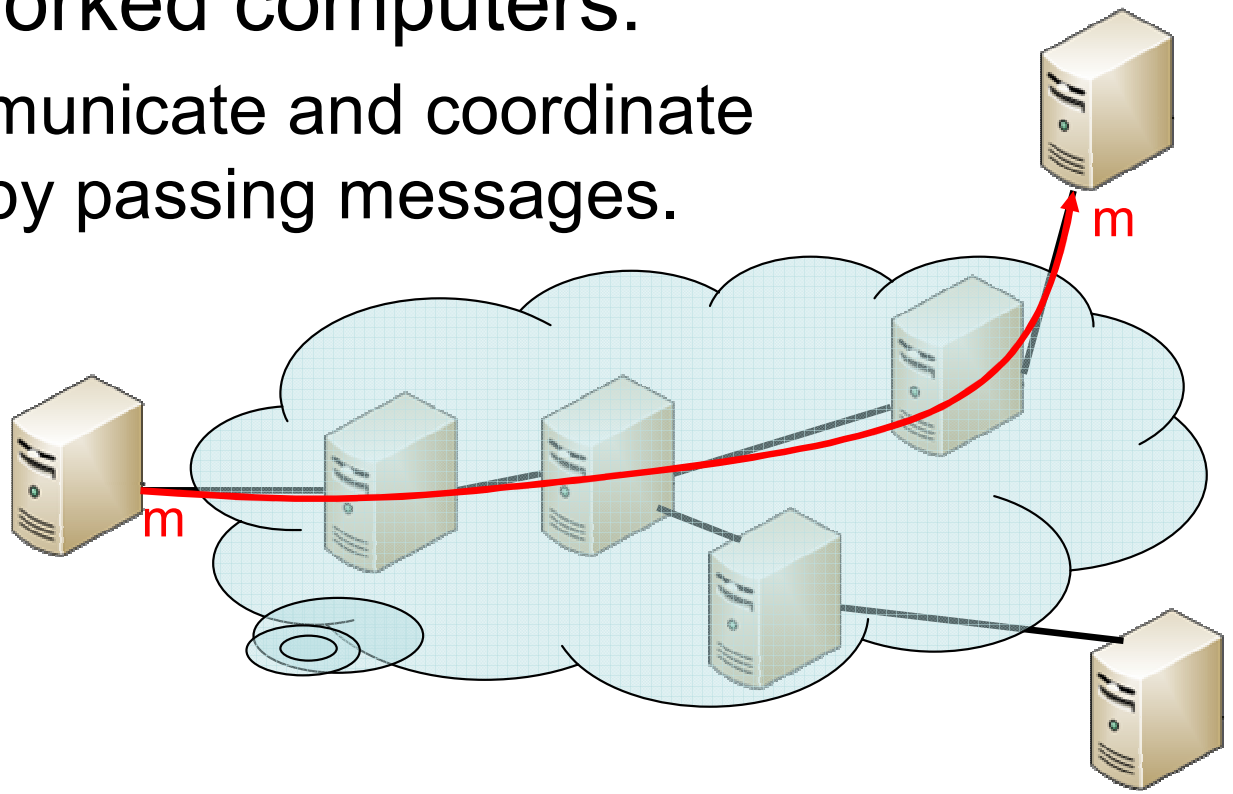
Faculty of Industrial Engineering,
Mechanical Engineering and Computer Science

School of Engineering and Natural Sciences

University of Iceland

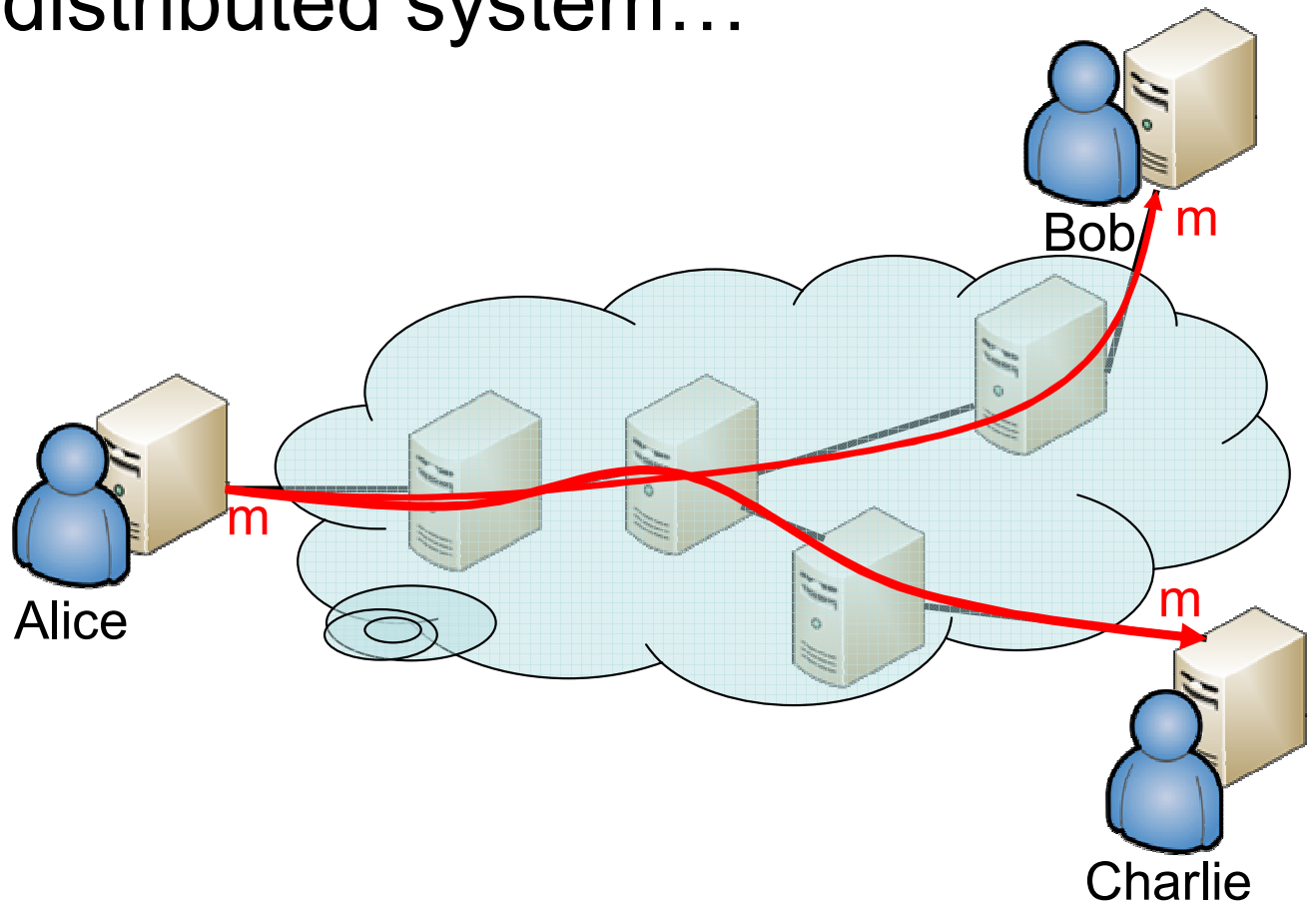
Introduction: Distributed Systems

- Hardware or software components located at networked computers:
  - Communicate and coordinate only by passing messages.



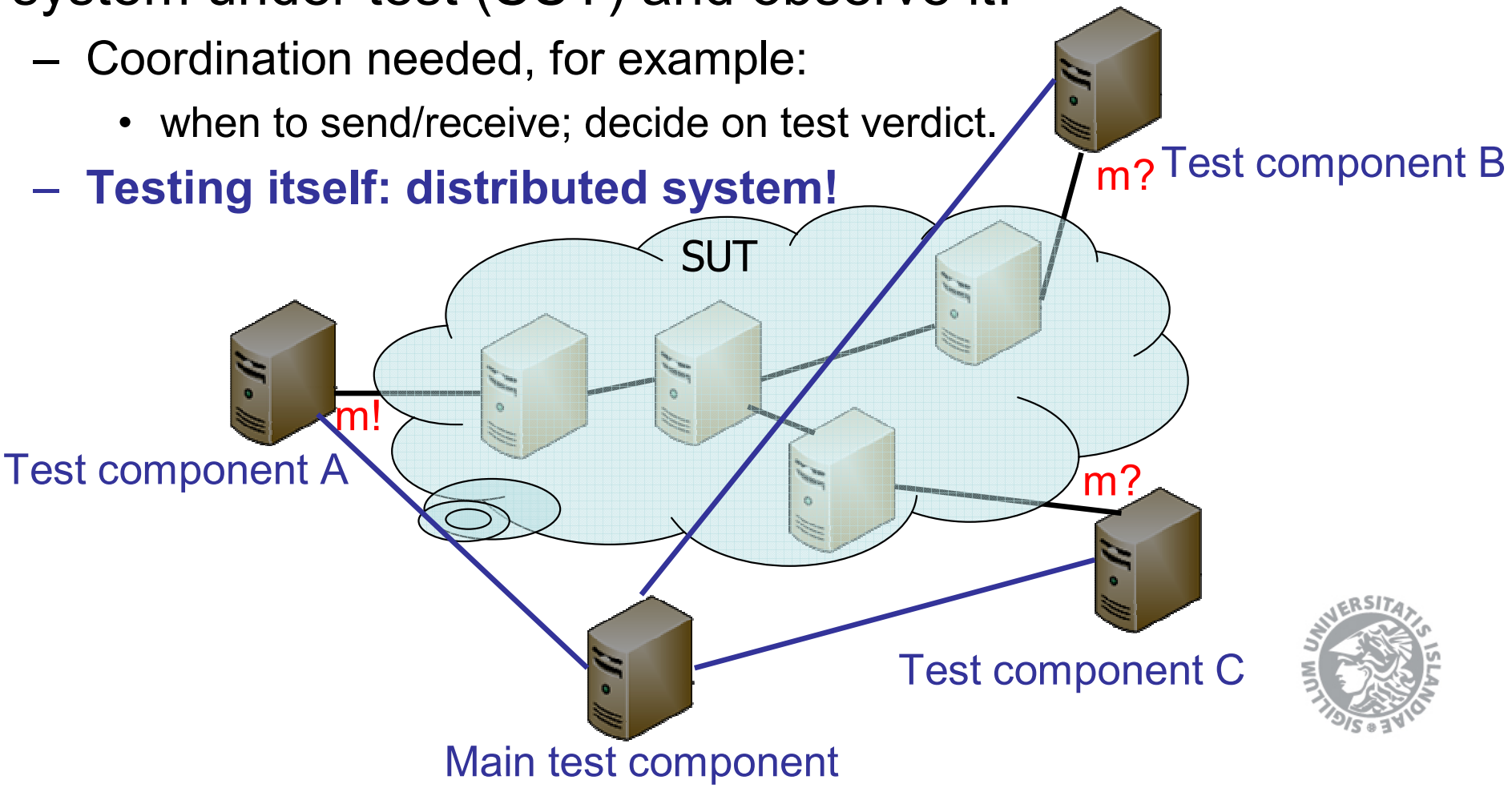
Introduction: Software Engineering – for example: Testing

- Test that a message *m* is correctly transmitted by a distributed system…

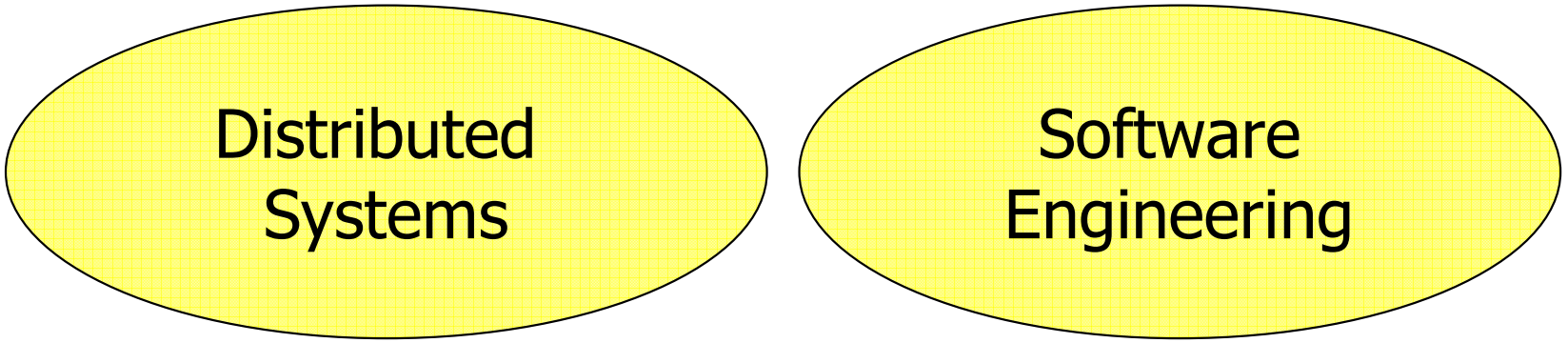


Introduction: Distributed Testing – How?

- Replace components by test components that stimulate the system under test (SUT) and observe it!
  - Coordination needed, for example:
    - when to send/receive; decide on test verdict.
  - **Testing itself: distributed system!**



My Research Fields



My Scientific Journey

- RWTH Aachen
  - Diploma Thesis: *Optimizing the Set of Selected Services in a CORBA Trader by Integrating Dynamic Load Balancing* (1999).


Source: RWTH Aachen

Thißen, Neukirchen: Managing Services in Distributed Systems by Integrating Trading and Load Balancing. ISCC 2000, IEEE.
Thißen, Neukirchen: Integrating Trading and Load Balancing for Efficient Management of Services in Distributed Systems. USM 2000, Springer.

Specification and Testing of Real-time Properties

- 2000–2003: EU FP5 “INTERVAL – Formal Design, Validation and Testing of Real-Time Telecommunications Systems”,
  - Uni Lübeck, Ericsson, France Telecom, Telelogic, Solinet, Teletel, Verimag.

- PhD thesis: *Languages, Tools and Patterns for the Specification of Distributed Real-Time Tests*, Uni Göttingen, 2004.

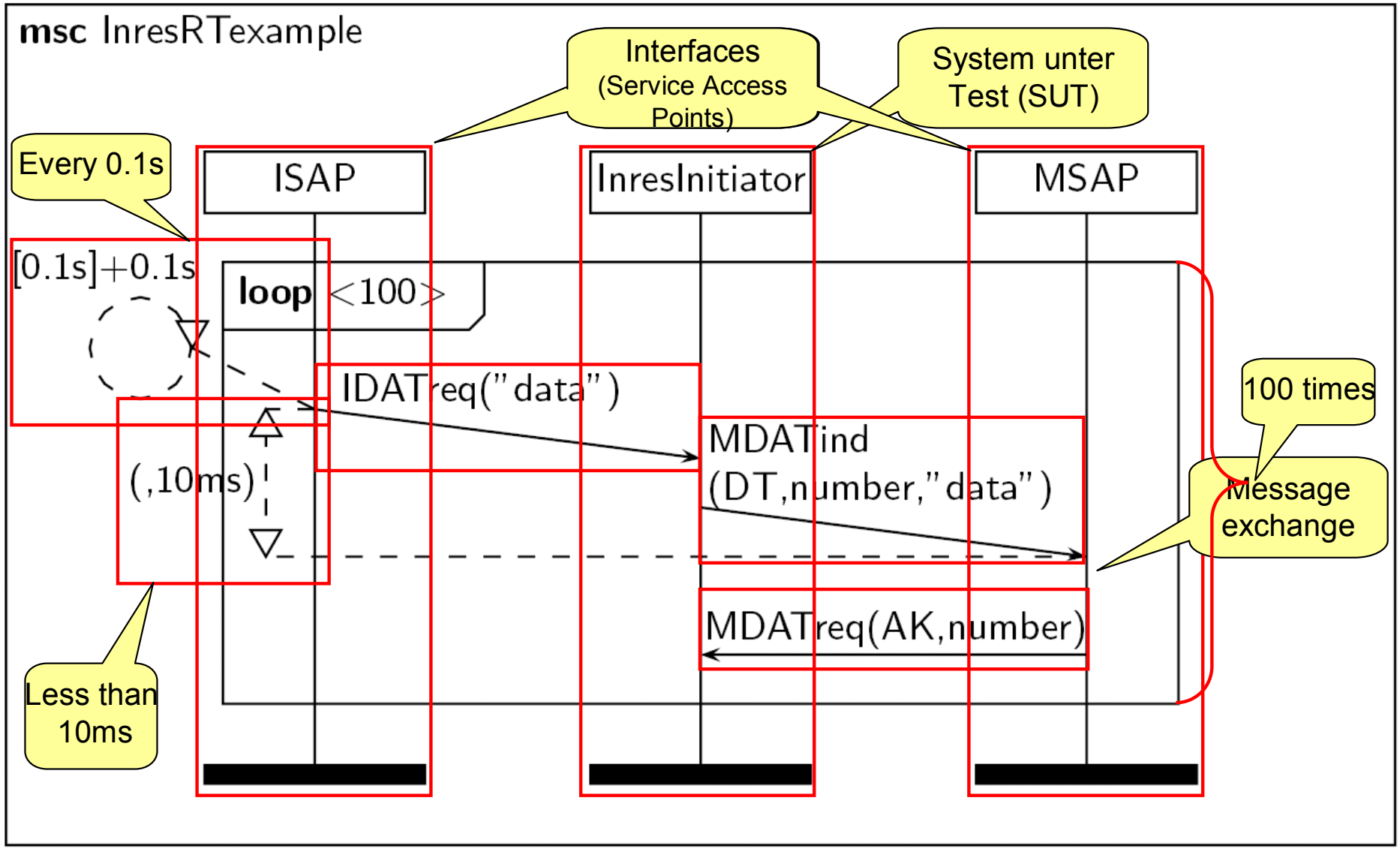
- 2008: “Testspezifikationstechnologie und -methodik für eingebettete Echtzeitsysteme im Automobil (TEMEA)”.
  - Fraunhofer, IT Power Consultants, Testing Technologies, Fourth Project Consulting (Uni Göttingen as sub-contractor).

Specification and Testing of Real-time Properties

- Add real-time concepts to distributed system specification & testing languages:
  - Message Sequence Chart (MSC),
    - International Telecommunications Union–Telecommunication Standardization Sector (ITU-T),
  - Testing and Test Control Notation Version 3 (TTCN-3),
    - European Telecommunications Standards Institute (ETSI).

- Participation in standardisation committees:
  - ITU-T, ETSI.

Example: MSC Describing Requirements on Periodic Stimulus and Latency



Example: Corresponding Timed TTCN-3 Test Case

```
testcase InresRTexample
{
  var float sendTime:=self.now+0.1;
  var float receiveTime;
  for (var integer i:=1; i<=100; i:=i+1)
  {
    resume(sendTime);
    ISAP.send(IDATreqType:{"data"});
    MSAP.receive(MDATindType:{DT,expected_num,"data"});
    receiveTime:=self.now;
    setverdict(evalLatencyOnline(sendTime,receiveTime,0.0,0.010));
    MSAP.send(MDATreqType:{AK,expected_num});
    sendTime:=sendTime+0.1;
  }
  setverdict(pass);
}
```

Read local clock

Resume execution at point in time

Time stamp for reception event

Evaluate time stamps

Calculate next point in time for next resume

Dai, Grabowski, Neukirchen: TimedTTCN-3 — A Real-Time Extension for TTCN-3. TestCom 2002, Kluwer.
Dai, Grabowski, Neukirchen: TimedTTCN-3 Based Graphical Real-Time Test Specification. TestCom 2003, Springer.
Neukirchen, Dai, Grabowski: Communication Patterns for Expressing Real-Time Requirements Using MSC and their Application to Testing. TestCom 2004, Springer.

Refactoring for TTCN-3 Code

- 2004–2008: PostDoc Uni Göttingen, e.g. teaching agile methods.
  - Refactoring: “A change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior.” (Fowler: *Refactoring–Improving the Design of Existing Code*, Addison-Wesley 1999)
- Developed catalog of systematic TTCN-3 code transformations.

  Zeiss, Neukirchen, Grabowski, Evans, Baker: Refactoring and Metrics for TTCN-3 Test Suites, SAM 2006, Springer.
- Since 2006: TTCN-3 Refactoring and Metrics tool (TRex).
  - Uni Göttingen & Motorola, on-going at Uni Iceland and Uni Göttingen,
  - Open source: **`http://trex.informatik.uni-goettingen.de`**
  - Also used in teaching, e.g. HBV101F Software Maintenance.

Neukirchen, Zeiss, Grabowski: An Approach to Quality Engineering of TTCN-3 Test Specifications. Int J Softw Tools Technol Transfer (2008) 10:309–326.
Baker, Evans, Grabowski, Neukirchen, Zeiss: TRex — The Refactoring and Metrics Tool for TTCN-3 Test Specifications. TAIC PART 2006, IEEE.
Neukirchen, Zeiss, Grabowski, Baker, Evans: Quality assurance for TTCN-3 test specifications. Softw Test Verif Reliab (2008) 18:71-79.

Example TTCN-3 Refactoring: *Inline Template Parameter*

- TTCN-3 example (unrefactored):

```
module ExampleModule {

  template ExampleType exampleTemplate(charstring addressParameter):={
    ipv6:=false,
    ipAddress:=addressParameter
  }

  testcase exampleTestCase() runs on ExampleComponent {
    pt.send(exampleTemplate("127.0.0.1"));
    pt.receive(exampleTemplate("127.0.0.1"));
  }

}
```

Example TTCN-3 Refactoring: *Inline Template Parameter*

- TTCN-3 example (refactored):

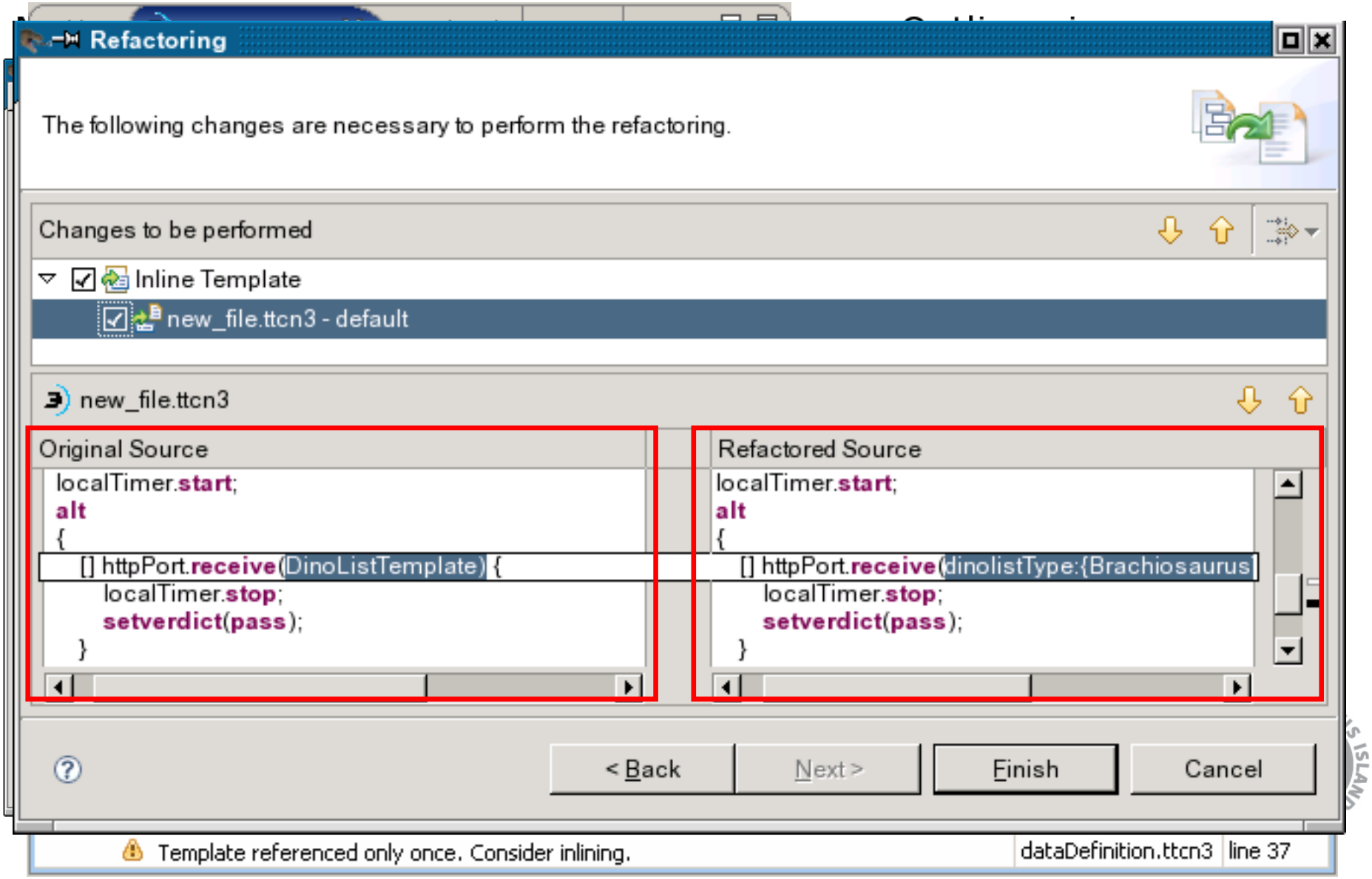
```
module ExampleModule {

  template ExampleType exampleTemplate:={
    ipv6:=false,
    ipAddress:="127.0.0.1"
  }

  testcase exampleTestCase() runs on ExampleComponent {
    pt.send(exampleTemplate);
    pt.receive(exampleTemplate);
  }

}
```

Development Environment TRex



Software Architecture and Quality

- University of Iceland.
  - 2008: Assistant Professor,
  - 2008–2014: Associate Professor,
  - Since 2014: Full Professor.

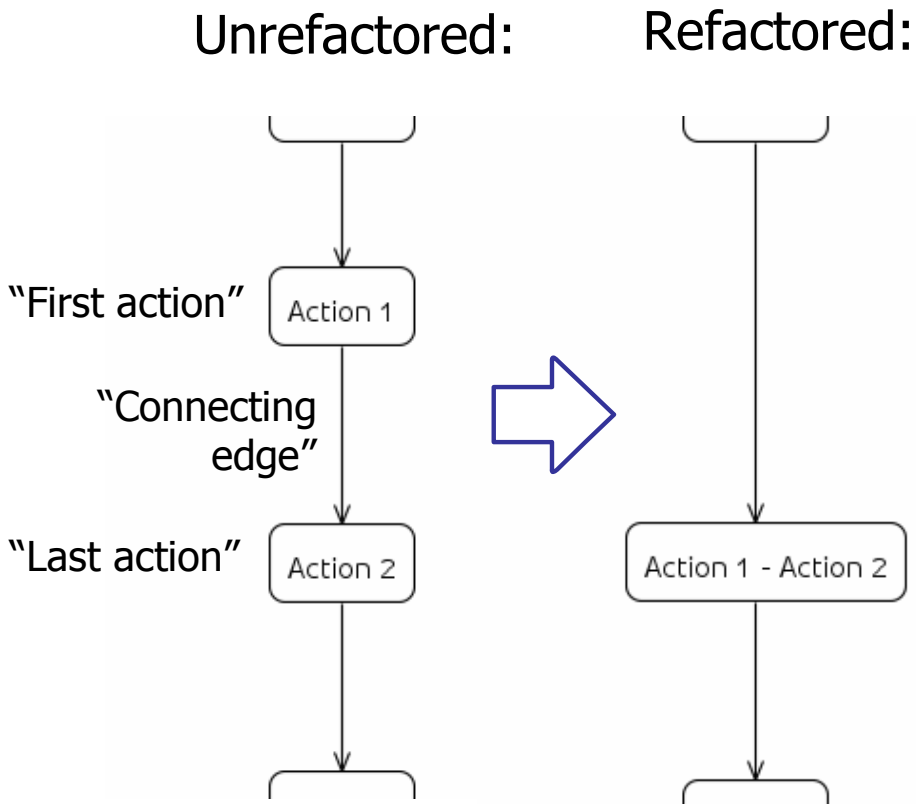
- Correlation of quality metrics and architectural metrics:
  - Avoiding extremes for the architectural metrics
    - e.g. neither extremely low nor high coupling of packages

  correlates with good quality
    - e.g. low number of reported bugs.

Hansen, Jonasson, Neukirchen: An Empirical Study of Software Architectures' Effect on Product Quality. J. Syst. Software 84 (2011) 1233–1243.

Refactoring for Unified Modeling Language (UML) – Example: Refactoring *Merge Actions*

- *Merge actions* of an activity diagram



- Steps:
  - Check if the last action has other incoming edges than the connecting edge each: if yes, refactoring is not applicable.
  - Move the target of all incoming edges from first action to last action.
  - Add name of first action to name of last action.
  - Remove the control flow edge that connects first and last action.
  - Remove first action.

UML Refactoring of UML Model and UML Diagrams

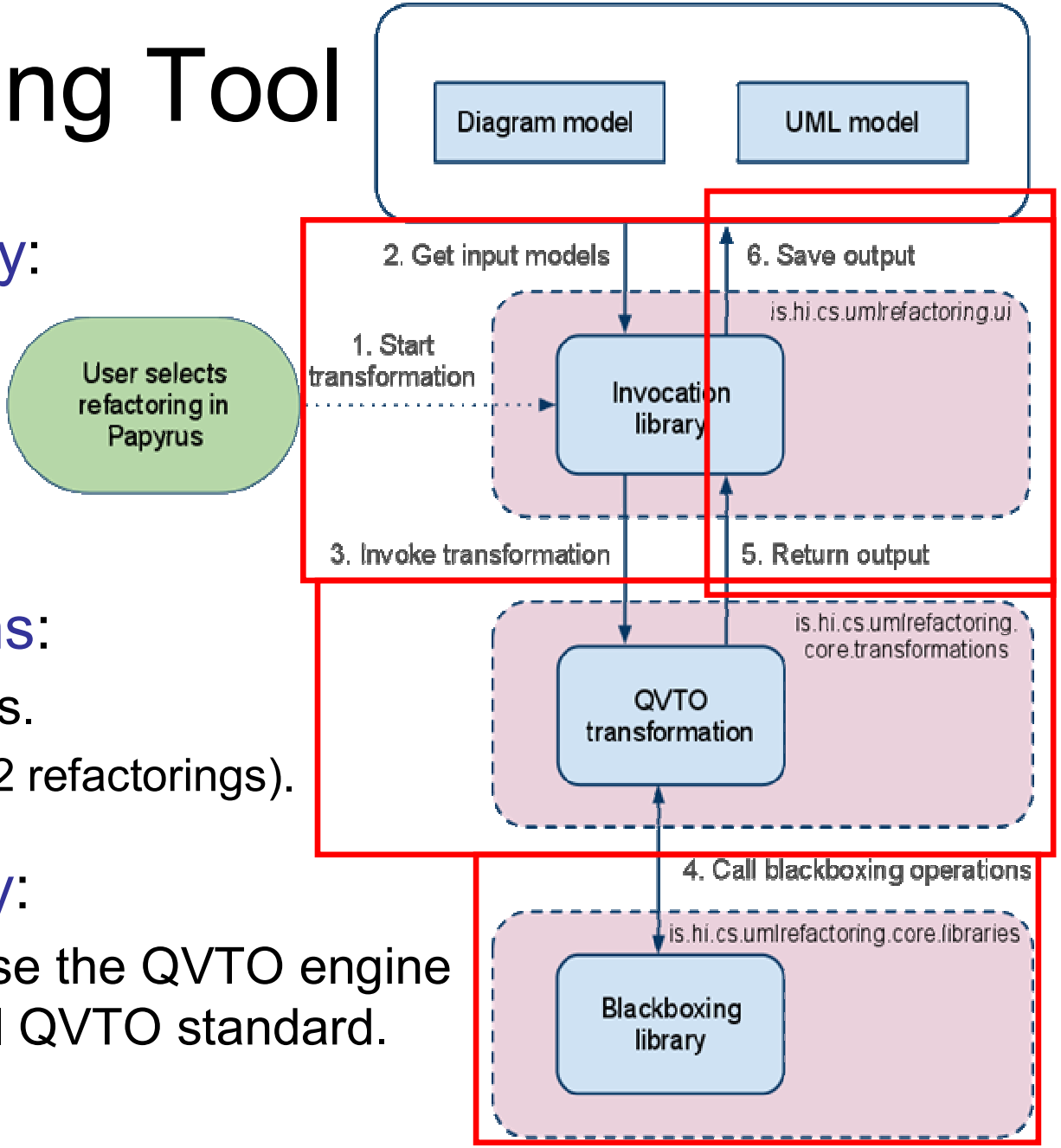
- UML model contains all elements and their relationship,
- UML diagrams visualises specific parts of the UML model from a particular viewpoint.
  - E.g. using a class diagram or activity diagram.
- Related UML refactoring work ignores UML diagrams:
  - Diagrams and model get out-of-sync after refactoring.

⇒ Approach: Refactor model + diagram using Model to Model (M2M) transformations.

  - M2M language: Query/View/Transformation Operational (QVTO).
  - Trick: Consider diagram as model,
  - Apply QVTO to model + diagram.

UML Refactoring Tool

- Java invocation library:
  - User interface + glue code to invoke QVTO transformations.
    - 284 lines of Java.
- QVTO transformations:
  - The actual refactorings.
    - 205 lines of QVTO (2 refactorings).
- Java black-box library:
  - Mainly needed because the QVTO engine did not support the full QVTO standard.
    - 57 lines of Java.



eScience/Grid Computing

- eScience: Computationally intensive science.
  - Typically distributed environment,
  - Series of computational or data manipulation steps (workflow).
- May involve Grid computing:
  Connected computing clusters from different organisations.
  - Highly heterogeneous.
- Grid workflow testing using TTCN-3.

Rings, Neukirchen, Grabowski: Testing Grid Application Workflows Using TTCN-3. ICST 2008, IEEE.

eScience/Cloud Computing

- Cloud computing: computational resources on demand, access via network.
  - Scale on demand,
  - Pay per use: no need to invest into hardware,
  - Virtualisation: no need to care about underlying hardware/heterogenity,
  - Cheaper.

- 2010: Northern Europe Cloud Computing (NEON):
  - Funded by the Nordic Data Grid Facility (NDGF),
    - Academic partners from Sweden, Norway, Denmark, Finland, Iceland.
  - Pilot projects to evaluate use of Cloud computing for eScience.

Edlund, Koopmans, Shah, Livenson, Orellana, Kommeri, Tuisku, Lehtovuori, Hansen, Neukirchen, Hvannberg: Practical Cloud Evaluation from a Nordic eScience User Perspective. VTDC11, ACM.

- Testing of Cloud computing applications using TTCN-3.

Tómasson, Neukirchen: Distributed Testing of Cloud Computing Applications Using the TTCN-3-based Jata Test Framework. NordiCloud 2013, ACM.

eScience/Big Data with MapReduce

- Big data: data too large to process it using traditional methods.
- Classical cluster computing:
  - Many CPU boards,
  - Central storage connected via some network to CPUs.
    - "Move data from storage to CPUs."
- MapReduce (e.g. Hadoop) with distributed storage (e.g. HDFS):
  - Many standard PCs: CPU with hard disk, network,
  - Distributed file system: each PC stores a part of the whole file system,
  - Processing is done on CPUs of those PCs where the data is stored.
    - "Move processing to where the data is."

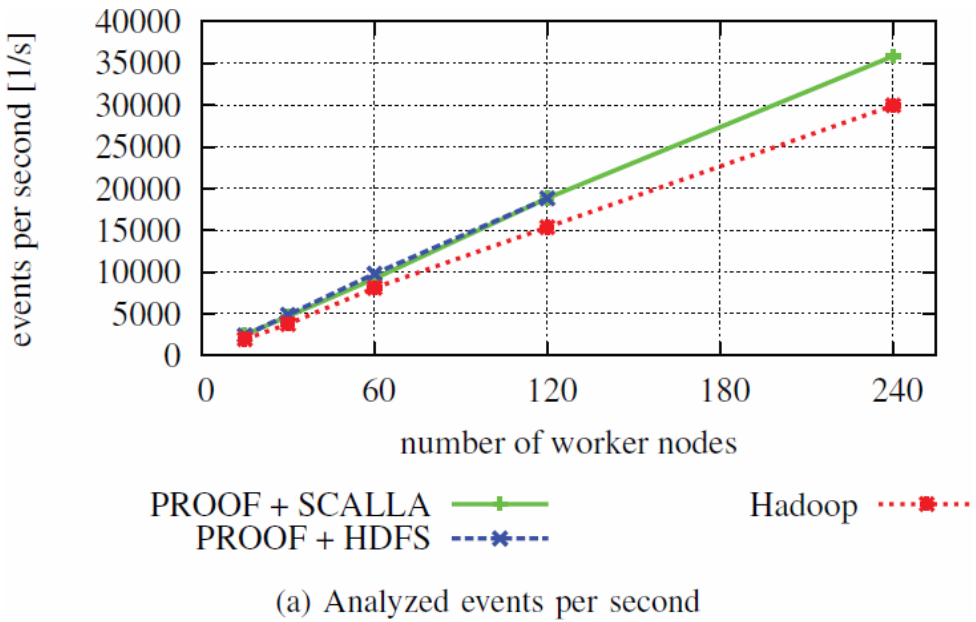
MapReduce for High Energy Physics (HEP)

- E.g. Large Hadron Collider (LHC) at CERN:
  - C++ library ROOT for easing HEP analyses,
  - For distributed/parallel computing: C++ framework Parallel ROOT Facility (PROOF) and distributed file system SCALLA.
  - Petabytes of data.

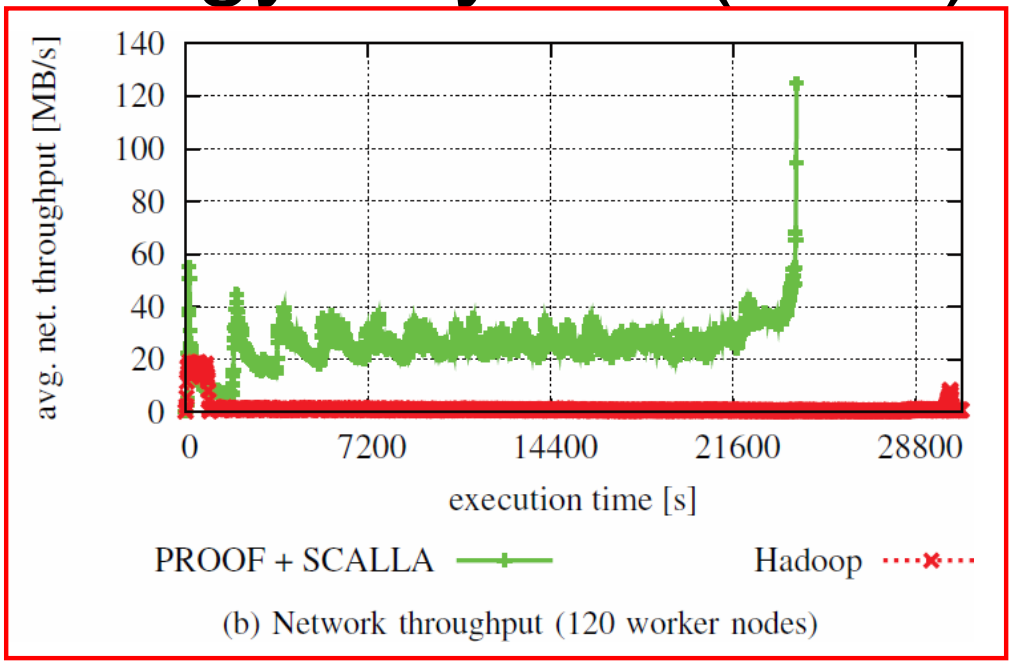
- Alternative approach using Hadoop/HDFS:
  - Sample HEP analysis provided by Max Planck Institute for Nuclear Physics,
  - Nordic High Performance Computing (NHPC):
    - Classic cluster setup with central storage.

  ⇒ Amazon Cloud computing resources to set-up Hadoop/HDFS cluster.
    - Research grant from Amazon.
    - Compared: PROOF+SCALLA, Hadoop+HDFS.

MapReduce for High Energy Physics (HEP)



(a) Analyzed events per second



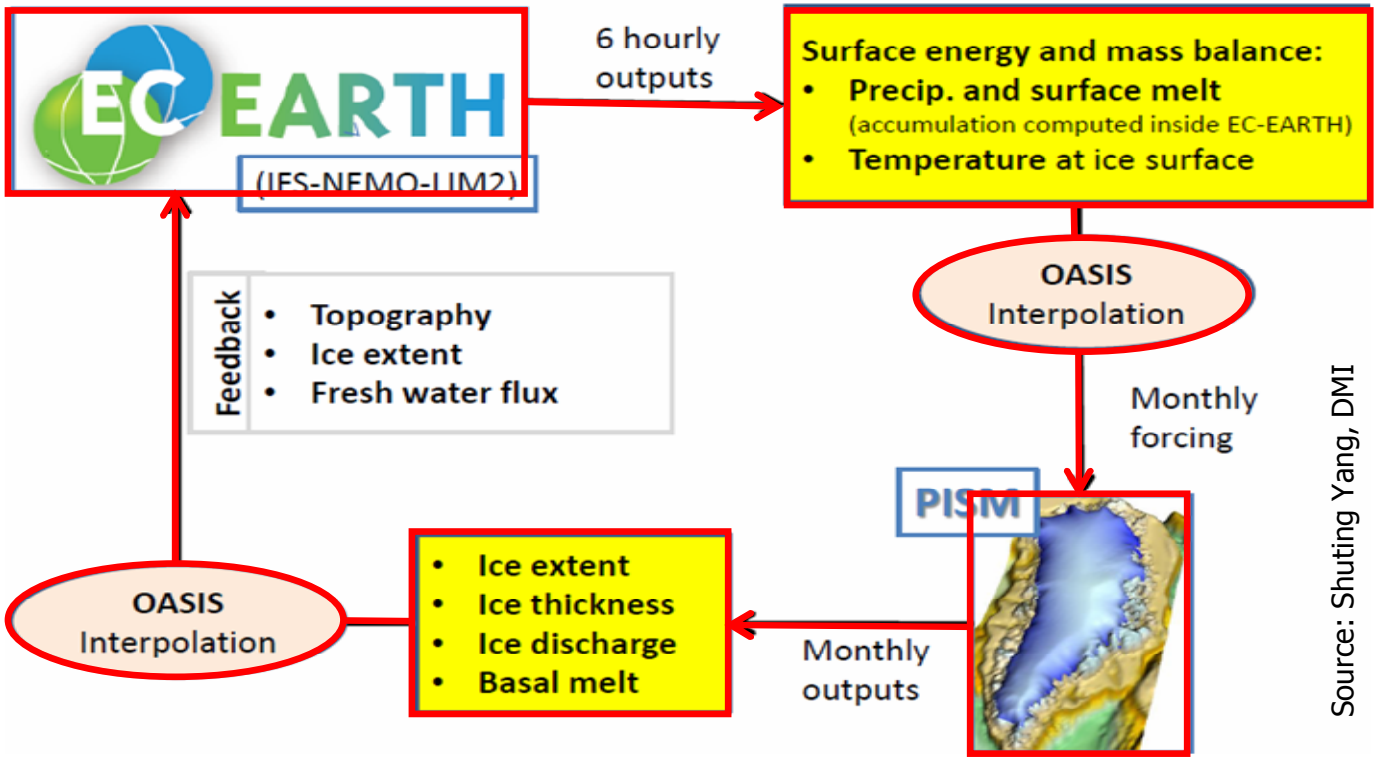
(b) Network throughput (120 worker nodes)

– Hadoop-based solution is slower than PROOF. Overhead due to Java, e.g.:
  – passing data between Java and C++ HEP analyses that use ROOT library.
\+ Hadoop/HDFS is more fault tolerant, wide-spread general purpose approach.
\+ Hadoop/HDFS significantly reduces network load (better locality).
  – Example HEP analysis is CPU intensive, I/O overhead not relevant.

Glaser, Neukirchen, Rings, Grabowski. Using MapReduce for High Energy Physics Data Analysis. MR.BDI 2013, IEEE 2013.
Schmelling, Britsch, Gagunashvili, Gudmundsson, Neukirchen, Whitehead: RAVEN – Boosting Data Analysis for the LHC Experiments. PARA 2010, Springer 2012.

Outlook: eScience Tools for Investigating Climate Change at High Northern Latitudes (eSTICC)

- NordForsk funded Nordic Center of Excellence.
  - 5 years, 13 research groups from all Nordic countries,
  - 25 million ISK funding for a PhD student at University of Iceland.
    Advertisement: In case you know good candidates: We are hiring!
    - Idea: Use workflow tools to couple currently isolated earth system models.



Thank you!

- I like to thank everyone who accompanied and supported me during my journey:
  - Teachers & supervisors,
  - Former and current colleagues,
  - Students & Research partners,
  - Friends,
  - My family,
  - Anyone I forgot to mention!