

Scalability

definition, assumptions and trade-offs



Reto Krummenacher, Elena Simperl (UIBK)
doug foxvog (NUIG)

January 8, 2008



TripCom at Web Scale



- The Web is *distributed, heterogeneous, dynamic, and open*.
- Required is a Triple Space platform which copes with these principles at Web scale.
- Classical guarantees encountered in closed environments cannot longer hold (per se).
- Adaptation of the *functional properties* publish and retrieve is required.
- Feasible *non-functional guarantees* must be determined from a scalability perspective

1. Scalability Deliverable
2. Scalability in TripCom
3. Recapitulation of non-functional / functional trade-offs
4. Configurations
 1. What is a configuration?
 2. How to define a configuration – some ideas!??

Scalability Deliverable

- First Final Draft: 31.1.2008
 - We need to know what there will be
 - We need to know what there will not be
 - Remember:
 - First final release submitted to EC by April'08
 - Internal deadline thus mid-March!

- Entirely missing so far
 - Chapter 4 (FUB, but also CEFRIEL, ONTO, NUIG)
 - Chapter 5 (WP4, 8a, and 8b)

- Evaluation Results (Chapter 6.2)
 - As soon as possible!!! The sooner the better...
 - Satisfiability of Section 6.1

PM for Scalability Task



- Total PM: 24
 - 2nd year (M13-M24): 14
 - 3rd year (M25-M36): 10

- PM per Partner:
 - TUW 5
 - UIBK 4
 - **FUB 4**
 - **USTUTT 4**
 - **ONTO 2**
 - NUIG 2
 - **CEFRIEL 2**
 - **Telefonica 1**

- chapter 1 Introduction
 - based on former chapter 3.1 (*The Scalability Task*)
- chapter 2 Scalability in TripCom
 - extended chapter 3.2 (*Definition*), additional arguments from the omitted chapter 2 (*Scalability in General*)
- chapter 3 Scalability Factors
 - former chapter 3.3/3.4
(*Scalability Factors/Assumptions&Trade-Offs*)
- chapter 4 Scalability Configurations
 - former chapter 3.5 (*Levels of Scalability*)
- chapter 5-8
 - former chapter 4-7

Scalability Deliverable



- 1 Introduction (UIBK, 31.01.2008)
- 2 Scalability in TripCom (NUIG/TUW, 15.09.2007)
- 3. Scalability Factors (UIBK, 15.09.2007)
 - 3.1 The Functional Aspects
 - 3.2 The Non-Functional Aspects
 - 3.3 Assumptions and Trade-Offs
- 4 Scalability Configurations (UIBK, 30.09.2007)
- 5 A Scalable Triple Space
 - 5.1 Triple Space Architecture (FUB/NUIG, 31.01.2008)
 - 5.2 Triple Space Conceptual Model (FUB, 31.01.2008)
 - 5.3 Triple Space Security (CEFRIEL, 31.01.2008)
 - 5.4 Triple Space Persistent Storage (ONTO, 31.01.2008)
- 6 Realization of Use Cases through Triple Space
 - 6.1 Semantic Web Services (USTUTT/UIBK, 31.01.2008)
 - 6.2 eHealth (CEFRIEL, 31.01.2008)
 - 6.3 EAI (Telefonica, 31.01.2008)
- 7 Evaluation of the Prototype
 - 7.1 Evaluation Procedure (TUW, 30.11.2007)
 - 7.2 Evaluation Results
 - 7.3 Discussion
- 8 Conclusions (USTUTT, 31.01.2008)

Scalability in TripCom

- The performance of a scalable tuplespace network would increase gracefully as additional spaces, kernels and users are added (assuming that additional resources are added as needed).
- Scalability measures would include data access time (latency) for the operations of TripCom, frequency and parallelism of such requests, database size, number of spaces, number of kernels over which a space is distributed, number of users, and the number of internal messages generated to route a request from a TripCom entry point to a kernel which hosts the required subspace.
- If one of these measures is increased while most of the rest are held fixed, the other(s) should increase in a graceful way.

- Some example scalability rules of thumb for TripCom follow:
 - TripCom data transmission time should scale no worse than linearly with respect to message size: $O(m)$. We do not intend to implement adaptive data compression schemes which could yield sublinear scaling.
 - The data access time should be highly scalable with respect to the number of kernels in the network: $O(\log n)$, although the number of kernels need only scale linearly with respect to the number of spaces: $O(n)$.

- Scalability of Triple Space

- NOT scalability of a TS kernel and its components
 - Are considered cheap operations
 - Are thus neglected in this discussion

- However, scalability models in TripCom
 - are more complicated than P2P models
 - demand more complex analysis and modeling methodologies than simulation
 - SIMULATIONS ARE IMPORTANT, AND MUST BE DONE AS SOON AS POSSIBLE
- What does it mean to have 10, 100 or 1000 spaces?
- What do we expect from 10, 100 or 1000 kernels?

Some examples:

- Increasing number of spaces per kernel:
 - 10 spaces on one kernel!
 - 100 spaces on one kernel?

- Increasing number of kernels with one space (assuming retrieval by URL):
 - 10 kernels each with one space!
 - 100 kernels each with one space?

- But, ...
 - request to multiple kernels
 - request to shared spaces

We therefore argue that proof of scalability by simulations might not be enough, nor really feasible, as there are ***too many parameters*** influencing the results!

- Tool for design time analysis of desires and requirements with respect to scalability

- Need for an ,offline‘ evaluation methodology
 - Determine scalability bottlenecks
 - Find alternative, still satisfiable and more scalable realizations

- We want to define the methodology to deliver such design time evaluations

Scalability Factors

- There are two dimensions to a middleware that influence its scalability
 - The exposed functionality - the service delivered to the clients - *the functional properties*.
 - The internal behavior of the system that is not obvious or visible to clients at interaction time - *the non-functional properties (NFP)*.

Functionals (1/3)

- Publishing information
 - `out(Triple t, URL space): void`
 - `out(Set<Triple> t, URL space, Time lease): void`
 - *Atomically writes a single triple into the space.*
 - *The operation makes no guarantee if and when the triple will be available in the space (unordered semantics).*
 - *The client is immediately free to perform further activities.*
 - *The client has to provide a resolvable URL which identifies a space.*

According Triple Space API, 5.11.2007

Functionals (2/3)

■ Retrieving information

- `rd(Template t, URL space, Time timeout): Set<Triple> s`
- `rd(Template t, Time timeout): Set<Triple> s`
- *Returns one match of the given template.*
- *The match may be a set of triples*
- *The operation makes no guarantee as to when the match would be returned to the client.*
- *A timeout is provided to give a temporal bound for returning a match. If no match has been found by the timeout period, an empty set is returned. This does not make any statement regarding the existence of a match in the space.*
- *No timeout can be specified by providing a null value.*

According Triple Space API, 5.11.2007

Functionals (3/3)

- *Query complexity*
 - Ranging from triple patterns $\langle :tripcom ?p ?o \rangle$ to arbitrarily complex queries.
 - Query resolution (reasoning).

- *Explicit vs. inferred data*
 - Publication at data level – no reasoning, multiple copies.
 - Retrieval at knowledge level – inference, materialization, only „one“ copy.
 - Local reasoning vs. distributed reasoning.

- *Transactions*
 - E.g. when publishing sets of triples within an atomic operation.

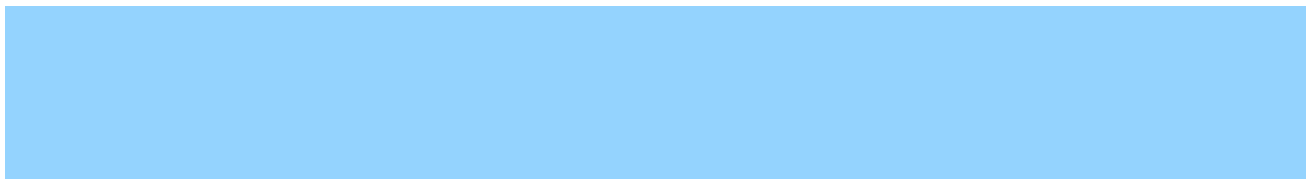
Non-Functionals



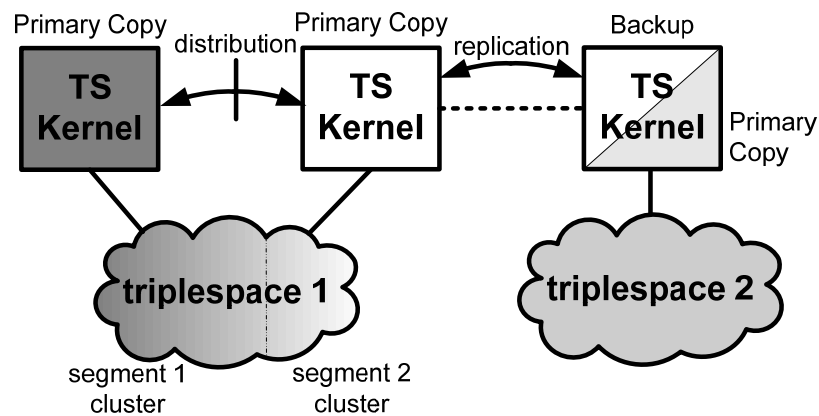
TS related
(dependability)



data/query
related



Technical Means for NFP



- *Redundancy*
 - duplication of functionality (n-out-of-m, $n < m$).
- *Recovery*
 - Re-installation of previous system state.
- *Load balancing*
 - decentralization of processes
 - by *replicating* data.
 - by *distributing* data (partitioning, clustering).
- *Synchronization*
 - eager, buffered, lazy...

- *Fault-Tolerance*
 - Redundancy: prevention, 1-out-of-2
 - Recovery

- *Availability*
 - Load balancing: 1-out-of-N

- *Reliability*
 - Redundancy: prevention, 2-out-of-3

- *Completeness*
 - No distribution of data

- *Consistency*
 - No replication: no system-caused inconsistency
 - Synchronization

- *Correctness*
 - No false-positives, a query engine issue, completeness

- *Durability*
 - Recovery: persistency is given by storage

- *Latency*
 - Load balancing

■ Scalability

- The trade-offs between *non-functional properties* and between *functionals* and *non-functionals* need to be analyzed in order to achieve a feasible level of scalability.
- An appropriate configuration for the desired behavioral and structural realization and the corresponding scalability level has to be selected.
- The configuration delivers a (limited) means to determine the expected degree of fulfillment for each of the properties.

Scalability Trade-Offs

Completeness \leftrightarrow Availability
Load balancing by distribution

Consistency \leftrightarrow Availability
Load balancing by replication

Consistency \leftrightarrow Fault Tolerance
Replication could be hidden

Reliability \leftrightarrow Response Time

Completeness \leftrightarrow Response Time

Consistency \leftrightarrow Response Time

Durability \leftrightarrow Response Time

Non-local writing results in latency

- Depending on the desired functionality and non-functionality, the space middleware has different possibilities to provide a scalable infrastructure.

The richer functionality the middleware provides, the less can be guaranteed about the non-functional properties and scalability.

Scalability Configurations

- Descriptions of the functional and non-functional
 - desires
 - requirements and
 - importance of each property

- A configuration must not describe a scalable realization
- Configurations deliver a means to analyse and evaluate (at design time) a desired implementation
- Existing configurations serve as guidance to the development of feasible solutions;
- i.e. configurations are written and applied by the developers and owners of spaces.

- Relevant non-functional properties

- (Service/TS) Availability
- Reliability
- Completeness

~~■ Correctness~~

- Consistency

~~■ Durability~~

- Latency

~~■ Scalability~~

Not tradable properties in TripCom:
- correctness of query engine
- durability of storage
- scalability of TripCom

- Security (orthogonal), Safety (kernel), fault tolerance (out of scope)

Configurations: The dimensions

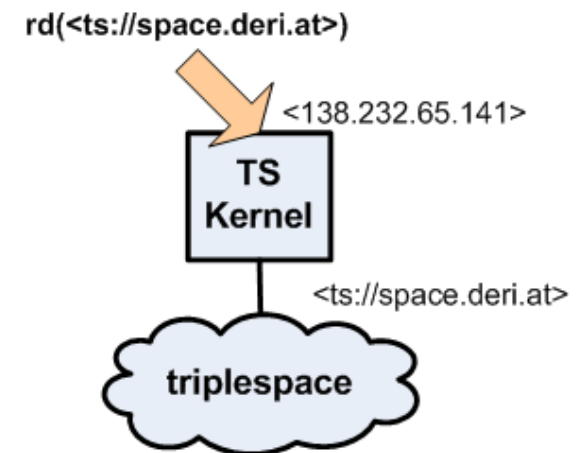


- Availability (Load Balancing)
 - Distribution
 - Replication
- Reliability (Redundancy: replication)
- Completeness
- Consistency
- Latency

- Coordination primitives
- Template matching / query complexity
 - Triple patterns
 - Complex RDF queries
- Transactionality
- Inference

Scalability Levels (1/3)

- Simplest (and most scalable) approach
 - The target space is indicated by a resolvable URL
 - Distribution is client-driven
 - Discovery by use of a DNS-like procedure
 - Ensures at least local scalability



Scalability Levels (2/3)

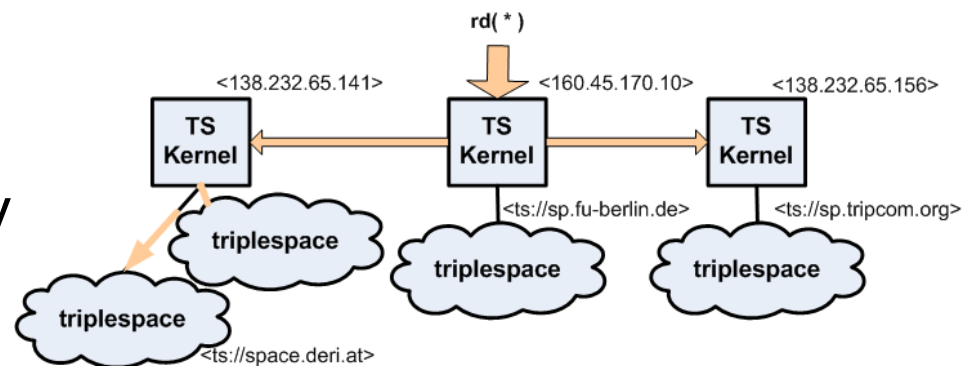
- Most complex approach – likely not scalable
 - Access to spaces without indication of a URL
 - (Knowledge-driven) retrieval on the entire Semantic Web
 - Query forwarding, distributed discovery of relevant spaces
 - Distributed reasoning to integrate distributed information sources/spaces or large scale (local) reasoning after transfer of triples

Scalability Levels (3/3)

- Loosening classical IR guarantees is of benefit to achieve scalability
 - Access to space(s) without indication of a URL
 - Best effort retrieval – no completeness
 - No guarantees about consistency
 - Rather data than knowledge-driven

- Performance

- Search scope
- Discovery strategy



- Further issues
 - Publication of sets of triples as atomic operation
 - **Issues with multiread/multiout**
 - Transactionality
 - Local transactions: data is visible to other hosts after commit
 - Transactions in case of distributed writing, i.e. publication at remote hosts
 - Locking a space is unfeasible for distributed, decentralized settings

- This is ongoing work!

- Some directions that we currently investigate:
 - Use the configurations (obviously)
 - Determine the required properties and the importance of the various factors

 - Potential techniques currently investigated:
 - Multi-Attribute Decision Analysis
 - e.g. Analytic Hierarchy Process (AHP)
 - Multi-Attribute Utility Functions
 - Sequential Decision Problems

Conclusions

- Scalability Deliverable
 - Due to January 31, 2008
 - Many open PMs to deliver!

- Scalability in TripCom
 - data access time (latency)
 - frequency and parallelism
 - number of spaces, number of kernels, number of users
 - number of internal messages

- Scalability in TripCom more complex than in traditional structured P2P overlays (DHT) (alternative realizations)
 - Functional and non-functional aspects
 - Trade-Offs

- Scalability Configurations
 - Descriptions of realizations with respect to the scalability factors
 - Applied for the design time evaluation and analysis

Thank you!