TripCom Kernel Architecture and Scalability Guidelines for Component Implementation

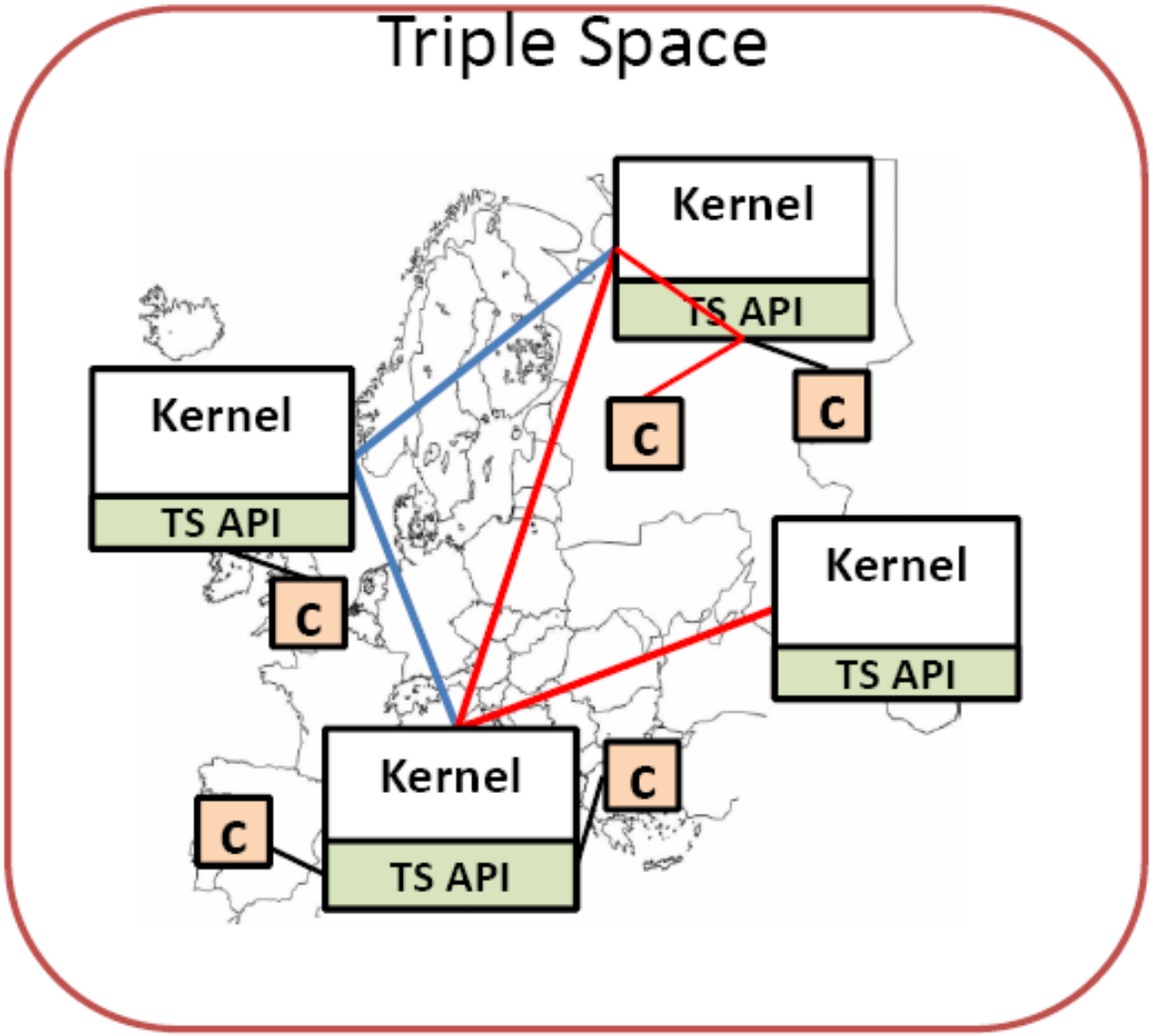
Agenda

- TripCom Kernel Architecture Recap
  - Global View
  - Triple Space Metamodel
  - Logical Kernel Architecture
  - Levels of Scale in TripCom
- Scalability Guidelines for Component Implementation

Sixth Framework Programme Priority 2
Information Society Technologies (IST)
Specific Targeted Research Project

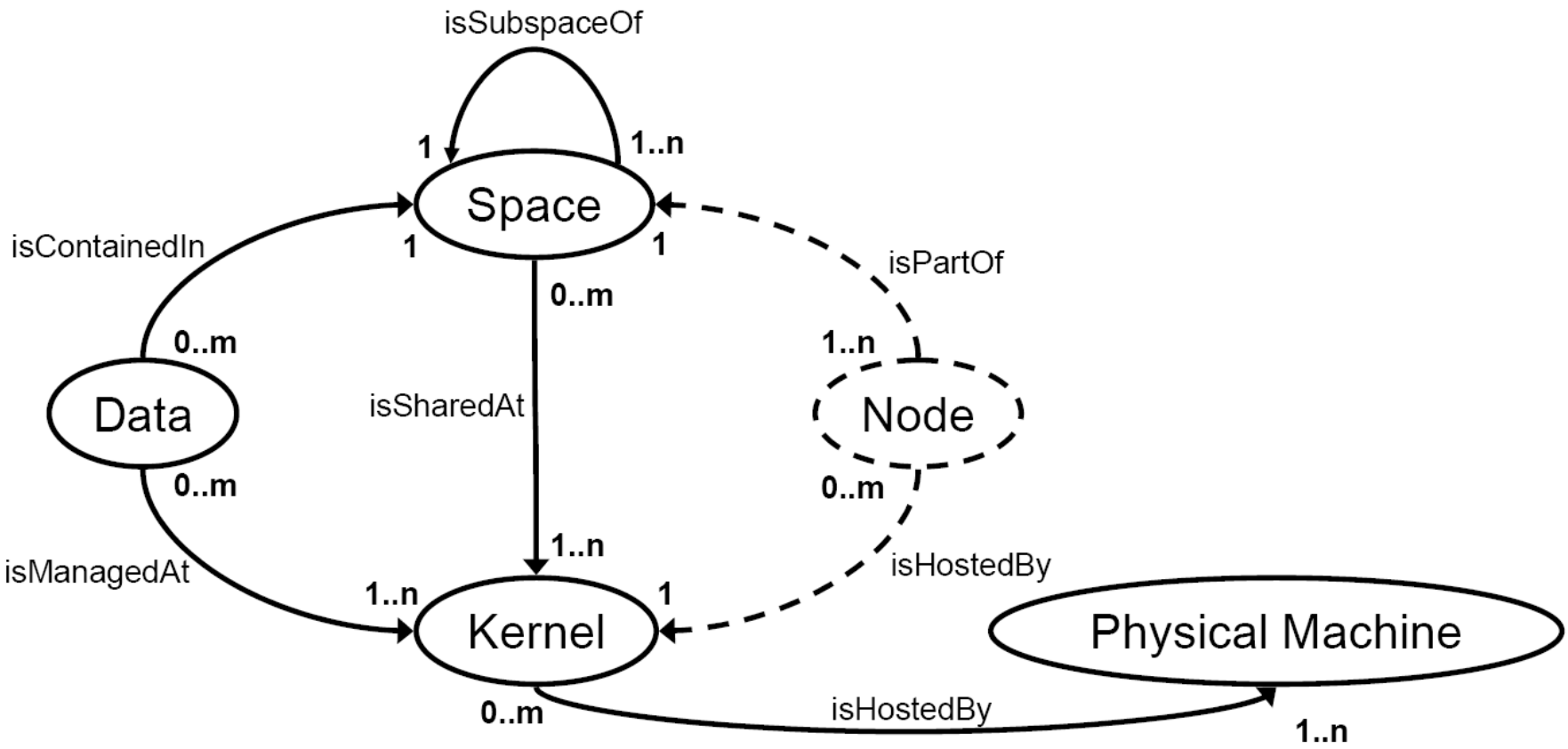
TripCom Architecture

Global View on Triple Space

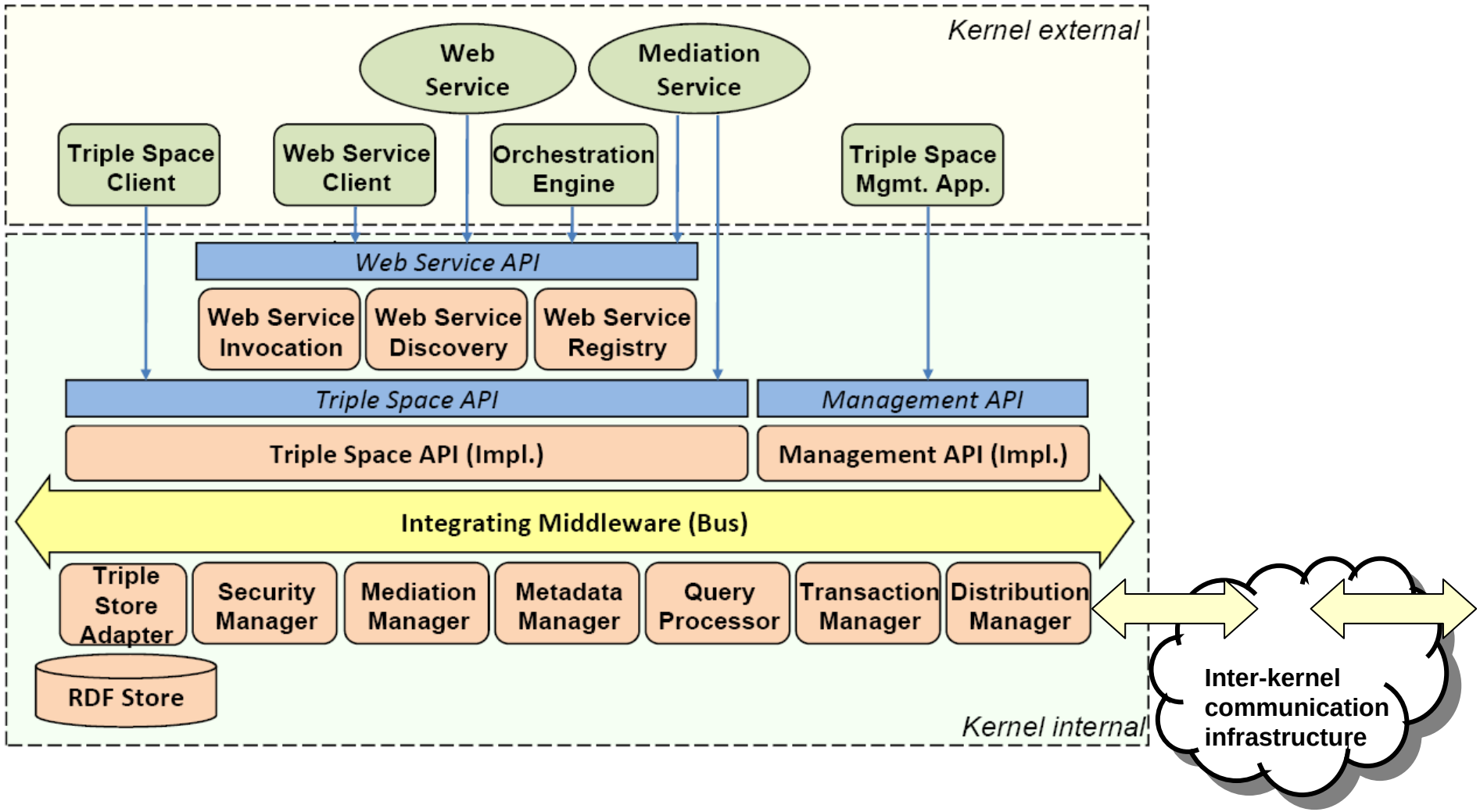


Sixth Framework Programme Priority 2
Information Society Technologies (IST)
Specific Targeted Research Project

TripleSpace Metamodel

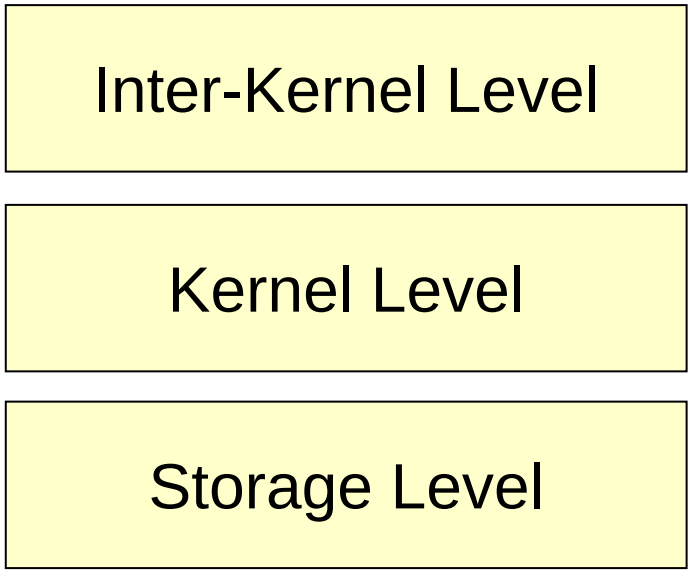


Kernel Logical Architecture



Levels of Scale

- Storage Level Scale – OWLIM
  - Allows connection to **multiple RDF stores** (planned for M24)
  - Best results on the LUBM benchmark
- Kernel-Level Scale – XVSM
  - Kernel Components can be **distributed over multiple machines**
- Inter-Kernel Scale
  - … will be discussed **in this meeting**



Scalability Guidelines for Component Implementation

(Frank Leymann)

Interpretation Rules for the Next Slides

- This is a very generic presentation of guidelines for the implementation of **scalable software**
- These guidelines apply for developing **products**
- TripCom will realize a **prototype**, i.e. some of the guidelines may be ignored
- Caveat: this subject is **very complex** and requires a lot of practical experience, i.e. TripCom cannot cover the full complexity

System Architecture

- Do not prevent **scaling out**
  - Potential use of more hardware
    - E.g. move from PC to cluster
- Do not prevent **scaling up**
  - Potential use of more resources on same system
    - E.g. partitioning of a database across multiple hard disks
- Use **logical layers**
  - Flexible distribution of subsystems (“engines”)
- Do not assume unnecessary **affinity**
  - Enables pooling & clustering (load balancing)

Sixth Framework Programme Priority 2
Information Society Technologies (IST)
Specific Targeted Research Project

Coupling and Cohesion

- Use **loose coupling** of building blocks
  - Enables distribution
- Use cohesion of groups of logically related building blocks
  - Only use late binding where necessary
    - E.g. for component interaction
    - Tight coupling / early binding within a component to be efficient

Concurrency

- Avoid thread creation on a per-request basis
  - E.g. use thread pooling instead
- Choose lock granularity carefully
  - Acquire locks as late as possible
  - Release locks as early as possible
- Use asynchronous execution for I/O bound tasks

State Management and Data Access

- Favor **stateless building blocks**
  - Avoids affinity, enables pooling
  - If needed: transfer state in messages
- In TripCom: Manage component state in the integrating middleware
  - Allows multiple instances of a component to share load

Communication

- **Avoid** interfaces that require **chatty communication**
  - E.g. by client-side validation, client-side caching, and batching of work
- **Avoid RPC** between your building blocks
  - Invoker is blocked
  - Use message queuing to decouple your building blocks
    - Enables "fire-and forget", and asynchronous calls
    - Reduces pain of long-running requests
- In TripCom: Use of space-based middleware for asynchronous communication

Deployment

- **Identify resource restrictions**
  - Processor power, storage capabilities,…
  - E.g. do not deploy a frequently used component on a weak machine
- **Take infrastructure restrictions into account**
  - Scalability of data store, machines,…
  - Consider network bandwidth restrictions
  - E.g. do not deploy a frequently used component on a poorly connected machine

Sixth Framework Programme Priority 2
Information Society Technologies (IST)
Specific Targeted Research Project

Conclusion

- Scalability on Level 1 and Level 2 is taken into account by the current TripCom architecture
- Guidelines have been proposed
  - “Scalability Guidelines for Component Implementation”
  - “TripCom Scalability Process”
- Level 3 scalability – the architecture of inter-kernel communication
  - To be defined in the next step (starting today ;-))

End of Document