The Mediator: What Next?

Talk by: Andy Cooke

Collaborators: Alasdair Gray, Lisha Ma, and Werner Nutt

Heriot-Watt University

Overview

- Next Steps:
  - All Insertables Should Stream
  - Choosing Query Plans.
- Future Steps:
  - Republisher Hierarchies?
  - Support More Queries?

All Insertables Should Stream

If a Producer publishes to a global table, it should be able to forward (stream) its table updates.

- **Archivers** will be able to **collect all updates** to a table, not just some
- **Consumers** can always **get full answers** from:
  - Archivers, as they are always complete
  - Complete Producers (no overlapping views)

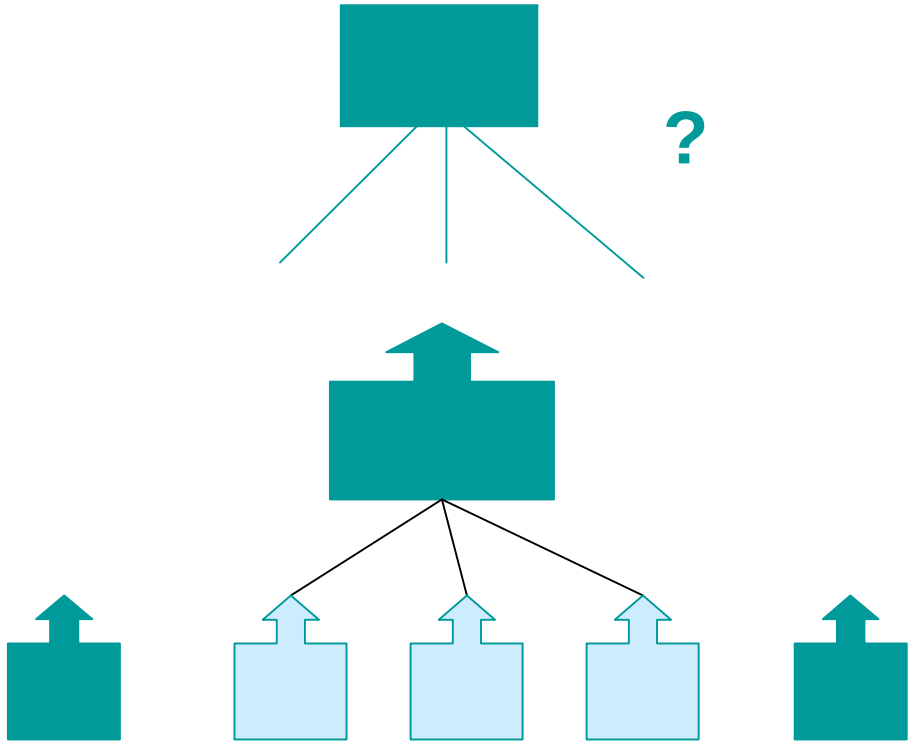
Simpler code, clearer semantics!

The Current Mediator

Latest Consumer

Latest Archiver

Primary Producers

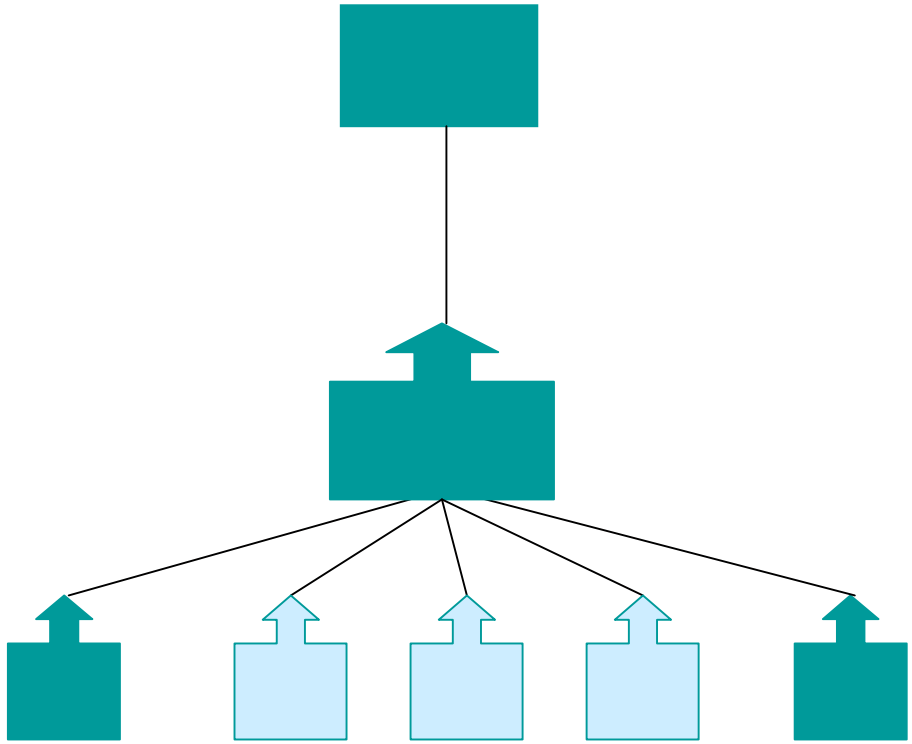


With a New Mediator

Latest Consumer

Latest Archiver

Primary Producers



Benefits to Users

- **Consistent Answers**, irrespective of where you are in the world!
  - Today, two primary LPs may offer different answers
  - …but a new mediator could prevent this
- **Improved Security**
  - Today, a “rogue” LP could be registered close to a Resource Broker, bringing down the Grid!
  - … but a new mediator would ignore it

Benefits to Users (cont.)

- **Full, Correct Answers**
  - As archivers will “fan-in” from **all** Insertables
  - Today, wrong answers may be returned to queries with aggregation!

We want to begin as soon as possible, in a separate CVS Branch

Discussion

We would like to add this fix to R-GMA before the end of the current project.

- It's needed before the mediator can be enhanced.
- Of course stability is the priority.

However,

- What other fixes are needed?
- How should these be prioritized?
- How do we organise their deployment?
- How do we minimise risk to stability?

Overview

- Next Steps:
  - All Insertables Should Stream
  - Choosing Query Plans
- Future Steps:
  - Republisher Hierarchies?
  - Support More Queries?

What is a Query Plan?

Query Plans are sent from the Registry to the Consumer. These should contain:

- Publishers that should be contacted, and
- Quality Description, e.g. COMPLETE flag

e.g. a one-time query with 3 plans (Archivers), two of which are complete.

e.g. a continuous query with one plan, involving 5 producers.

Which Query Plan should be used?

If a Consumer Agent has a choice of Query Plans, which should it choose to execute?

- The plan that returns the *fastest* answer?
- The plan that returns the fastest, *most complete* answer?
- The plan offering the *freshest* tuples?

… or should users have a say?

Which Query Plan is fastest?

Fastest Query Plan could be found by:

- *Measuring the time* it takes for a getStatus() message to return.
- This measurement could be made for every new plan:
  - When the *Consumer registers*, and
  - When *Consumer is notified* of new Producers

Monitoring Completeness

- Registry maintains completeness flags for all Publishers
- Registry informs Consumers whenever a Publisher's *status changes*
- Registry monitors status of producers:
  - Primary Producers are complete if there are **no overlapping** producers
- Archivers monitor their own status
  - An Archiver is complete when it has **fully started**
  - It tells the registry when this happens.

Choosing Query Plans

- If there are several complete plans, which one is the best?
- If all possible plans are incomplete, which one is the best?
- Can it be that an incomplete plan is better than a complete one?

Incomplete plans could be ranked by **counting primary keys**

- Easy for latest archivers
- More difficult for history archivers!

Choosing Query Plans

We propose an algorithm that involves:

- Archivers tell registry when they've fully started (i.e. have contacted all SPs in its plan).
- Consumers maintaining a "league table", ranking plans according to:
  - Their "closeness"
  - Their "completeness"
- Primary keys can be counted to decide between two incomplete plans

Overview

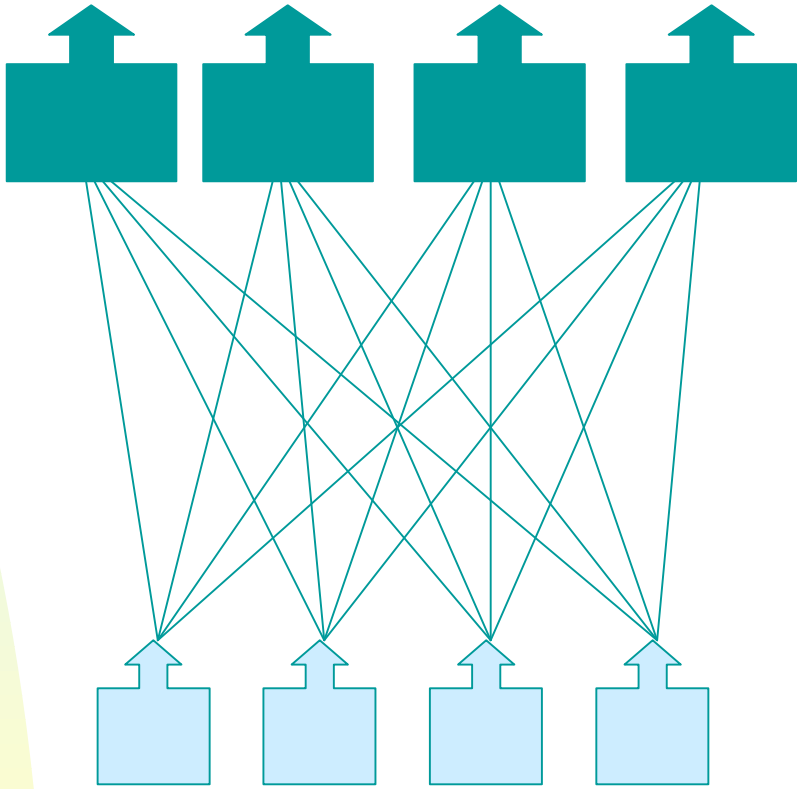
- Next Steps:
  - All Insertables Should Stream
  - Choosing Query Plans
- Future Steps:
  - Republisher Hierarchies?
  - Support More Queries?

Republisher Hierarchies

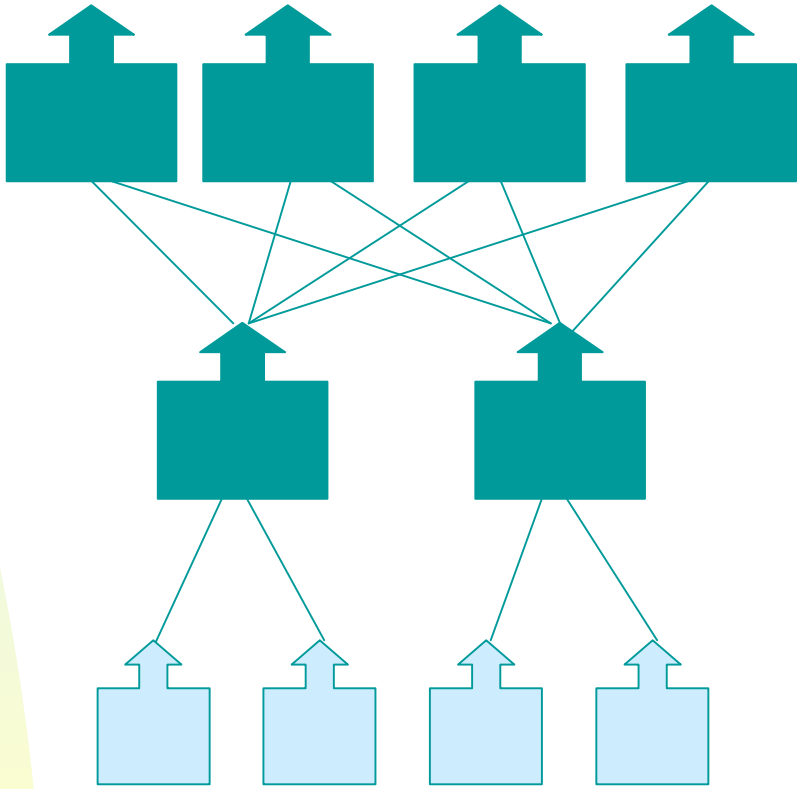
Republisher Hierarchies may help to:

- Reduce network traffic
- Improve the max republishing rate
  - less threads!
- Share load across publishers
  - as more choice for consumers.

One layer of archivers



Two layers of archivers



Less traffic, less load?

Would Republisher Hierarchies help?

- We need measurements:
  - How many Cs can a P serve?
  - How many Ps can a C stream from?
  - Max insert rate into a P?
  - Max republishing rate?
  - where is the bottle neck?
- Would the schema support hierarchies?
- Would the registry support hierarchies?

Overview

- Next Steps:
  - All Insertables Should Stream
  - Choosing Query Plans
- Future Steps:
  - Republisher Hierarchies?
  - Support More Queries?

Supporting more Queries

Improve language for continuous queries?

- Queries with OR
- Queries with aggregation, e.g.

"average over the last minute"

Support more one-time queries?

- When Archivers have partial views
- When no Archivers and need to merge?