GRID COMPUTING IN R

WITH REDIS MESSAGE QUEUES

Jonathan Adams

R in Insurance 2016

BACKGROUND

- Common for me to run lots of jobs in R
  - Fitting complex models
  - Running simulations from a CAT model
  - Numerical integration of posterior distributions

BACKGROUND

- First used `snow` package.
- Worked as follows:
  1. Bundle code into function
  2. Create cluster of worker nodes
  3. Spread your jobs among the nodes before processing
  4. Wait for it to finish or throw an error

BACKGROUND

- Pros
  - Run multiple jobs simultaneously
  - Multiple methods of cluster creation
  - Could load-balance some clusters
- Cons
  - Not all cluster could be load-balanced
  - Common to be waiting for one or two nodes to finish
  - Poorly handled errors could cause loss of all work
  - Reproducing errors was complicated
  - Results not available until all processing finished
  - Could not increase size of cluster mid-processing

WAYS TO IMPROVE

- Allow quick scaling up and down
- Use any number of servers
- Automatically load-balance
- Access results as they are available
- Separate errors for easy access and reproducibility

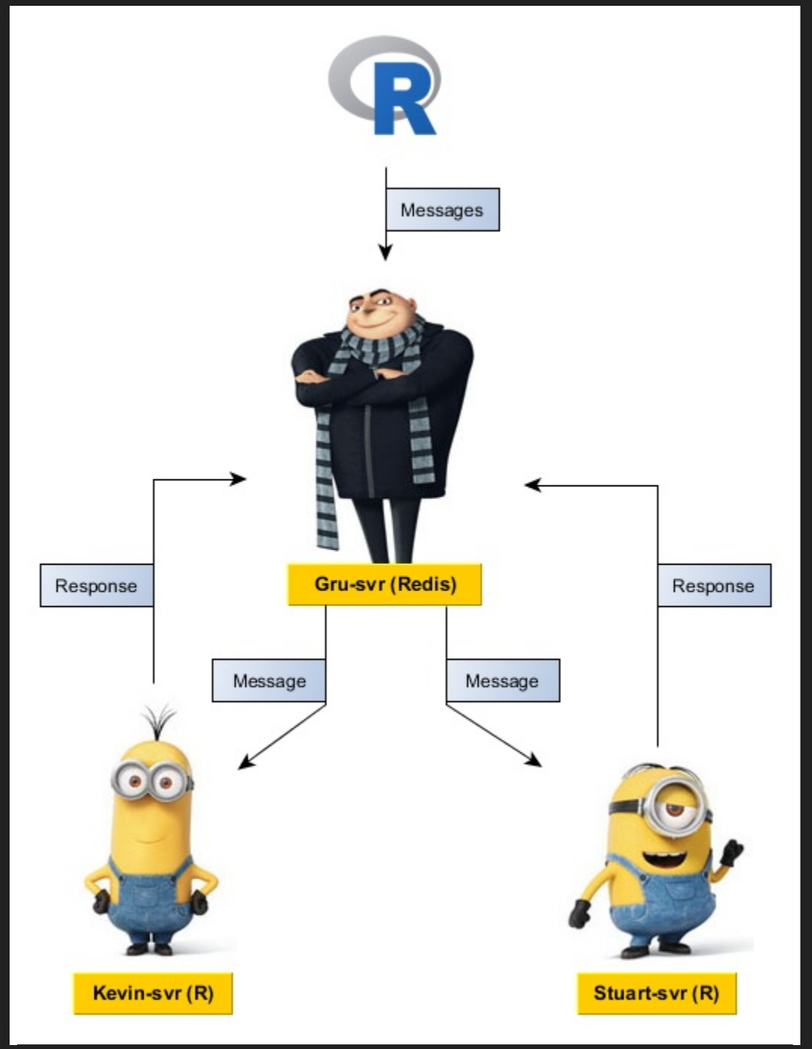
REDIS

- Redis is an in-memory NoSQL database
- All objects are strings
- Allows clients to wait for messages to be available
  - Messages are queued and clients are first-come-first-served
  - Pop from and push to queues just like an array
- Bindings made available in R through `rredis` package
  - Stores any R object by serializing

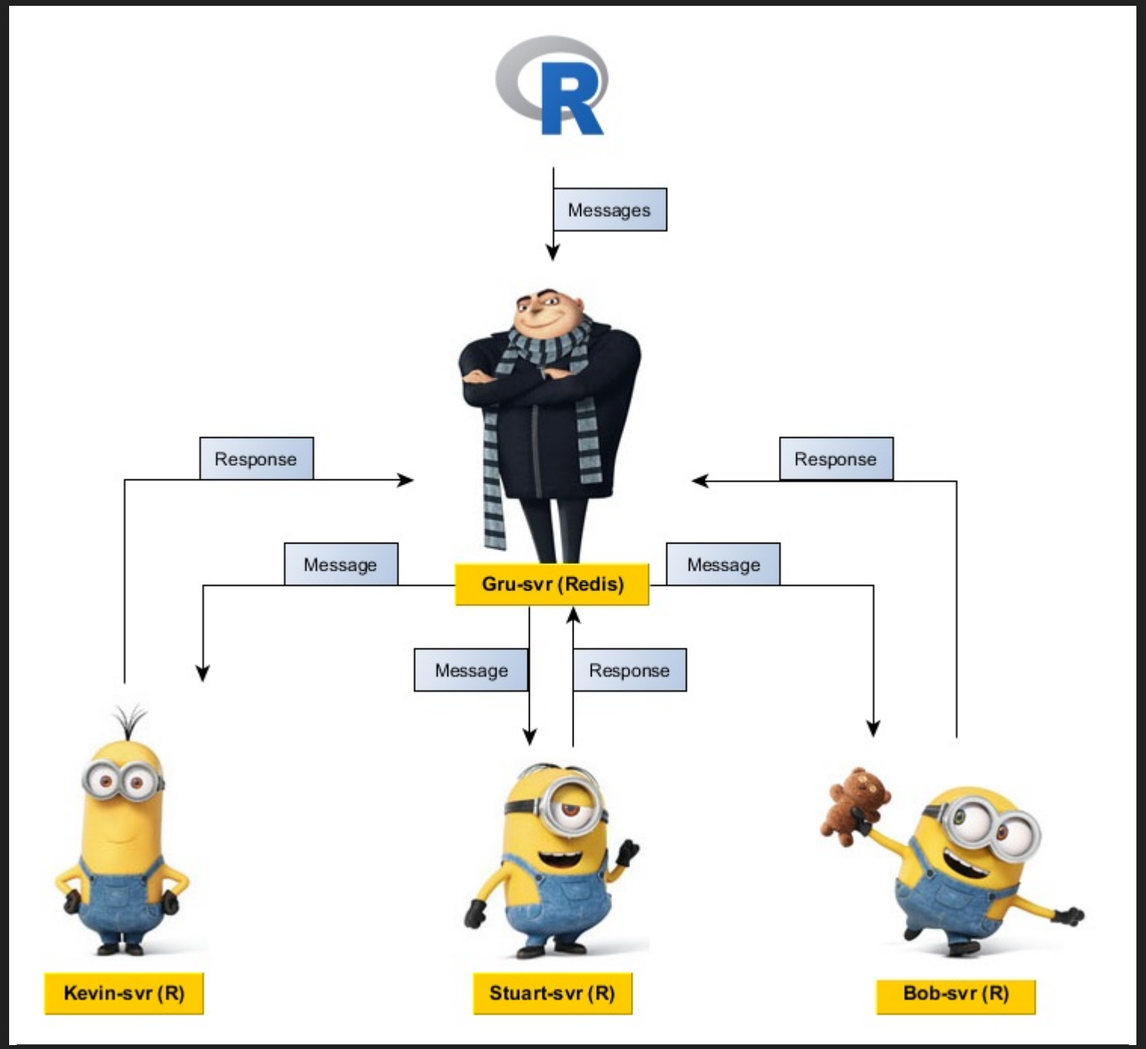
MESSAGE FORMAT

- Messages will be lists with the following keys:
  - Function - A function definition with one argument, `params`
  - Parameters - A list of all parameters needed to execute the function
  - ResultsQueue - Name of the queue to store results in
  - ErrorQueue - Name of the queue to store errors in

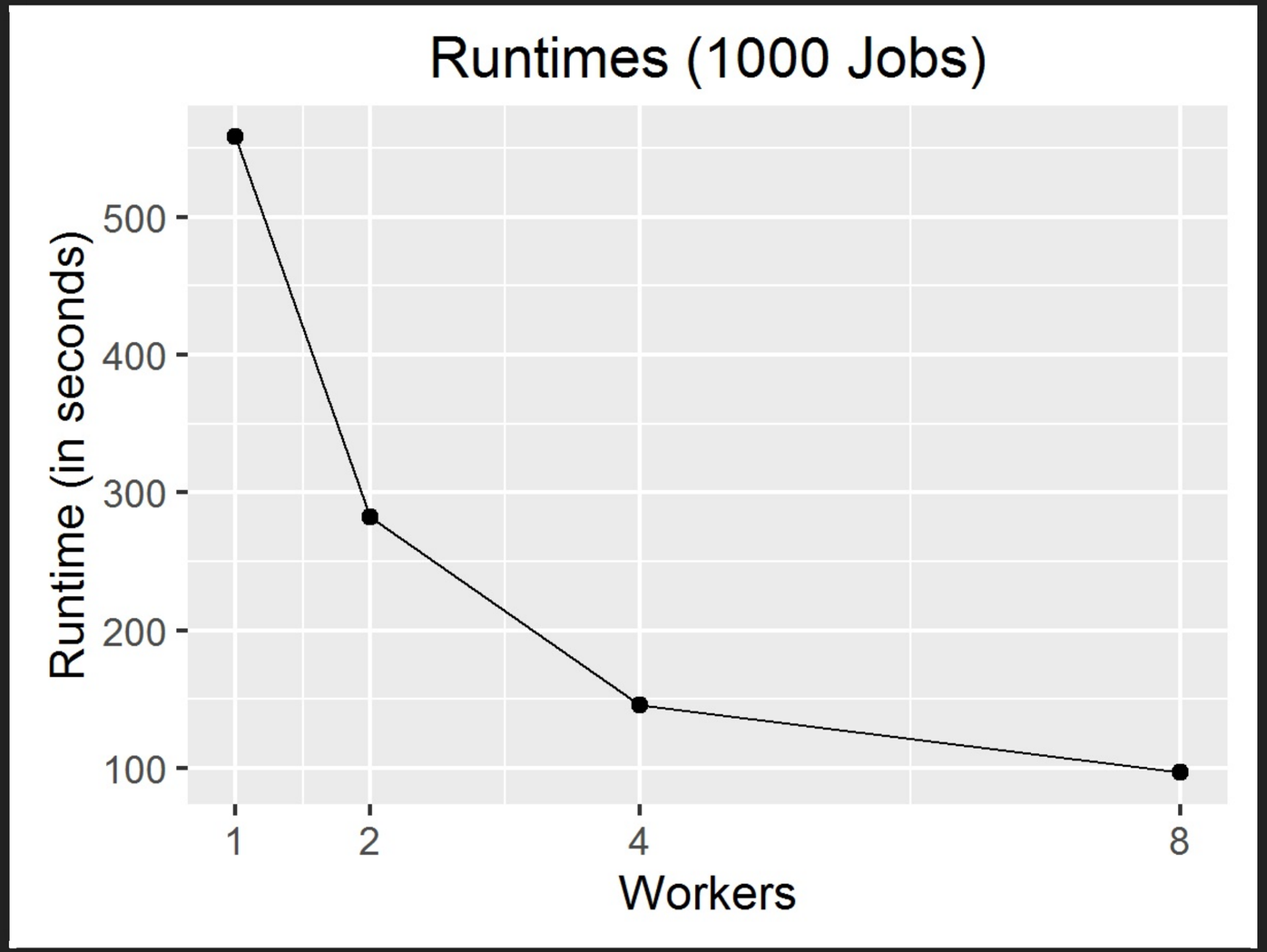
FLOWCHART


Messages
Response
Gru-svr (Redis)
Response
Message
Message
Kevin-svr (R)
Stuart-svr (R)

FLOWCHART


Messages
Response
Response
Message
Message
Gru-svr (Redis)
Message
Response
Kevin-svr (R)
Stuart-svr (R)
Bob-svr (R)

RUNTIME


Runtimes (1000 Jobs)
Runtime (in seconds)
500
400
300
200
100
1
2
4
8
Workers

CONCLUSION

- Improvements revisited
  - Allow quick scaling up and down
  - Use any number of servers
  - Automatically load-balance
  - Access results as they are available
  - Separate errors for easy access and reproducibility

CONCLUSION

- Available on GitHub https://github.com/PieceMaker/rminions
- Easily deployed in the cloud via Docker
- Can communicate with all servers via PUB/SUB
- Can update worker to accept requests from other languages
- Similar package: `doRedis`