Analysis of Remote Execution Models for Grid Middleware

Andrei Hutanu, Stephan Hirmer, Gabrielle Allen, Andre Merzky

Center for Computation & Technology

AT LOUISIANA STATE UNIVERSITY

Introduction

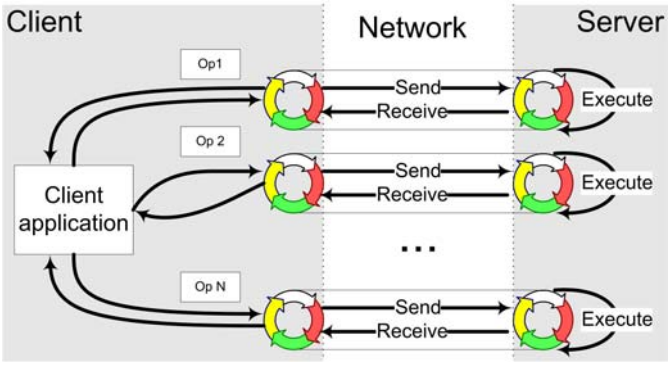
- Performance deterioration due to latencies of remote operations
  - Most relevant when two entities have multiple rounds of communications
    - Examples : copy multiple files using a data transfer service, access various sections of a remote data object for visualization

SAGA

- Low-level communication paradigms require performing latency-hiding techniques in the application
- High-level APIs abstract the communication layer
  - Example : SAGA. GGF effort for simple API for utilizing grid services
  - Need to transparently include latency hiding, be flexible in their latency hiding techniques

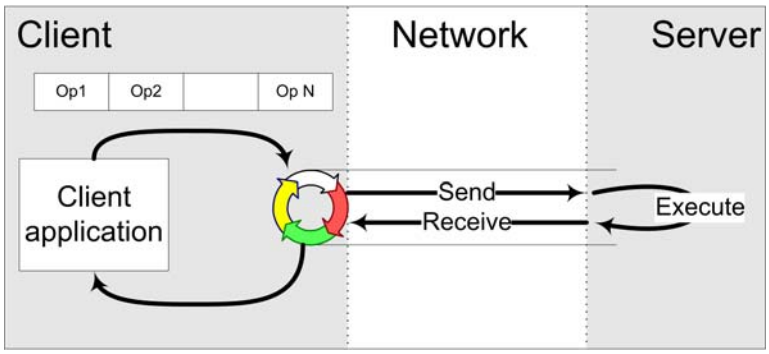
Asynchronous model

- Using threaded execution to hide remote latency : each operation
  spawns a thread
- Usual concurrency issues. Ordering not preserved.
  Server should accept multiple connections



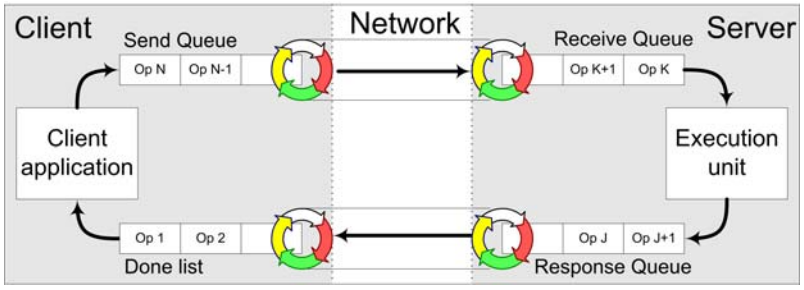
Bulk model

- Multiple operations sharing common semantics are combined into a single remote invocation
- Operations must start at the same time. Bulk interface needed on the server



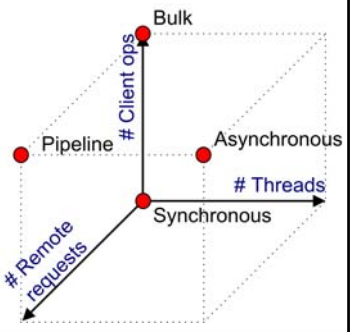
Pipeline model

- Client-server system has three segments
- Requests/responses sent over a persistent connection using a dedicated thread
- Server implementation prescribed. Ordering ok



Execution models

- Synchronous : *one* operation, *one* request single thread
- Bulk : *n* operations, *one* request, *one* thread
- Asynchronous : *n* ops, *n* requests, *n* threads
- Pipeline : *n* ops, *n* requests, *k* << n threads



Performance model : synchronous

- Typical programming model, operations are synchronized.

$t_{sync}(n) = n * t_{sync}(1)$

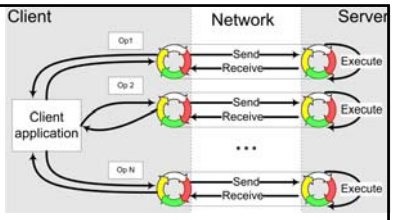
$t_{sync}(1) = t_{server\_op} + t_{comm\_sync}$

$t_{comm\_sync} = t_{lat} + message\_size / bandwidth$

(here t_{lat} includes network RTT and other per-message overhead and is independent of the message size)

$$t_{sync}(n) = n\left(t_{lat} + \frac{message\_size}{bandwidth} + t_{server\_op}\right)$$

Performance : asynchronous



- Communication time for each channel

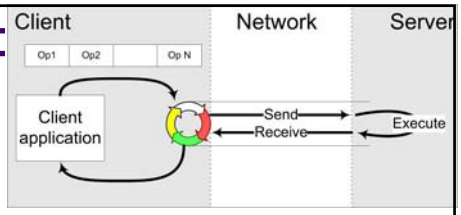
$$t_{comm\_async} = \frac{t'_{lat}}{n_{net-||}} + t_{thread} + \frac{message\_size}{bandwidth \times n_{net-||}}$$

  - t'_{lat} now also includes connection set-up time and authorization
  - n_{net-II} is a network speed-up factor given by the usage of multiple threads

$$t_{async}(n) = n\left(t_{comm\_async} + \frac{t_{server\_op}}{n_{server-||}}\right)$$

  - $n_{server-II}$ is the speed-up factor on the server

Performance: bulk



- Main optimization : one request for *n* ops.

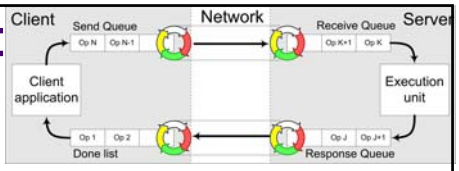
$$t_{comm\_bulk}(n) = t'_{lat} + \frac{n \times message\_size}{bandwidth}$$

- Latency occurs only once. Message size could be smaller

$$t_{bulk}(n) = t'_{lat} + n\left(\frac{message\_size}{bandwidth} + t_{server\_op}\right)$$

- Execution time could also be optimized

Performance: pipeline



- Consider the generic case (k segments)

$$t_{pipeline}(n) = (n-1)\, max_{i=1..k}(t_{segm}[i]) + \sum_{i:=0}^{k} t_{segm}[i]$$

- For our 3 segments:

$$\begin{aligned} t_{send}(1) &= \frac{t_{lat}}{2} + \frac{request\_size}{bandwidth/2} \\ t_{execution}(1) &= t_{server\_op} \\ t_{receive}(1) &= \frac{t_{lat}}{2} + \frac{response\_size}{bandwidth/2} \end{aligned}$$

- Separate request and response but bandwidth also additive

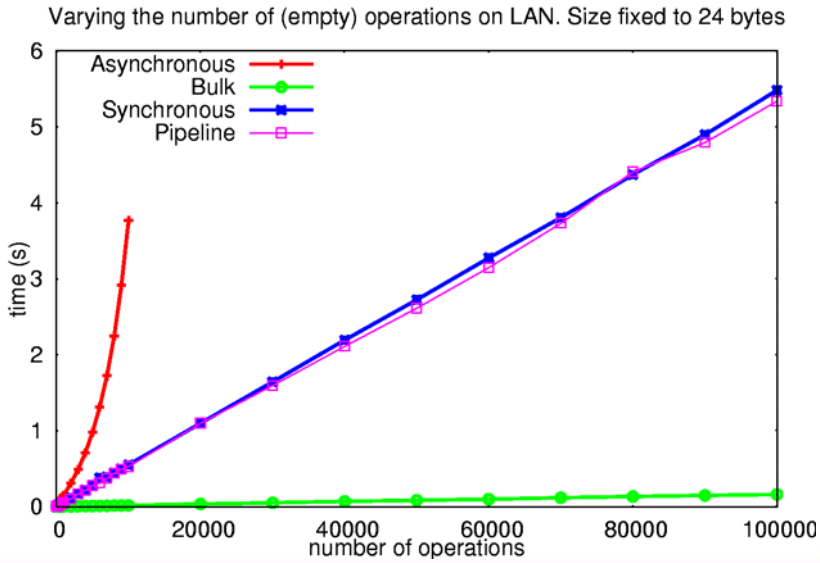
Benchmarks

- As in the models, operations of equal size
- Two networks
  - Direct fiber connection (5Gbps throughput, 0.1 ms RTT) – LAN
  - Internet (7 Mbps server->client, 40 Mbps client->server, 40 ms RTT) – WAN
- Two operation types
  - NOOP : empty operation, server deliver data from a zero buffer
  - FAOP : remote file access : client specifies the offset and size of a remote read, server delivers data from a file

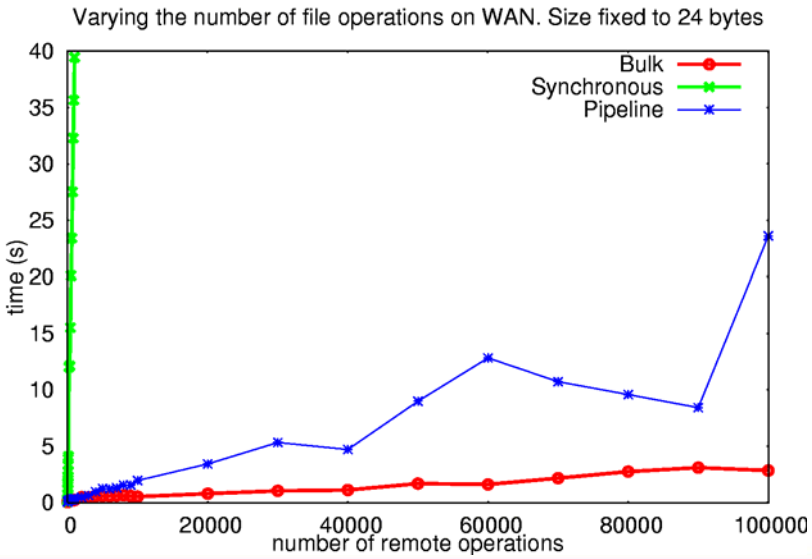
Per-operation overhead

- The first benchmark keeps the size of the operations small and varies their number
  - Indicates per/operation overhead independent of operation size

LAN : bulk best



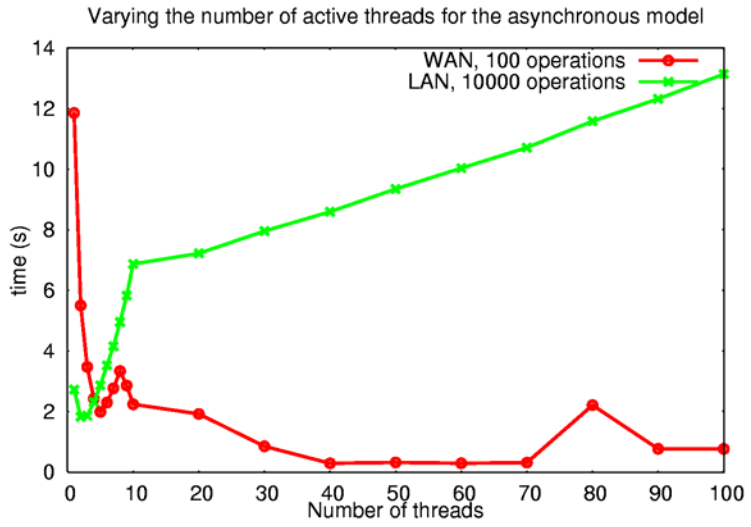
WAN : synchronous falling behind



TCP considerations

- For the asynchronous model, multiple threads => parallel connections => increased throughput.
  - Iperf shows a speedup of 1.2 on the LAN and 1.7 on the WAN is achievable
  - However, too many threads will damage performance
  - Need to find the balance point (only way to limit number of threads is to limit the number of operations)

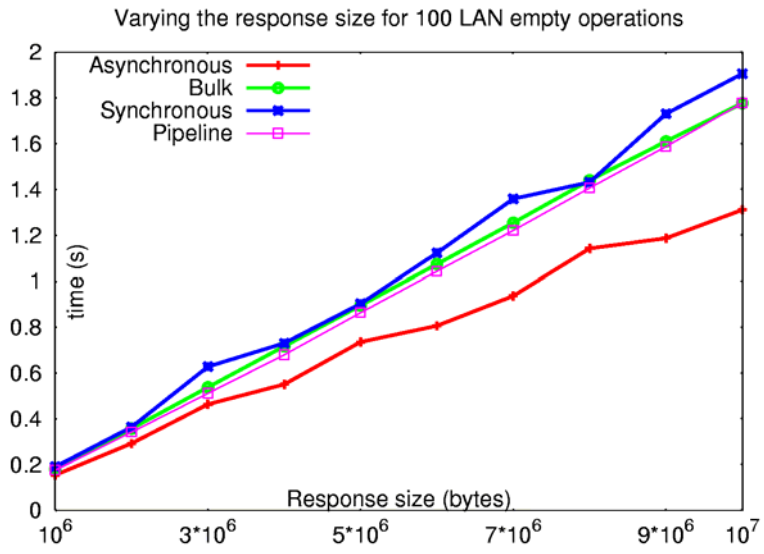
Async model



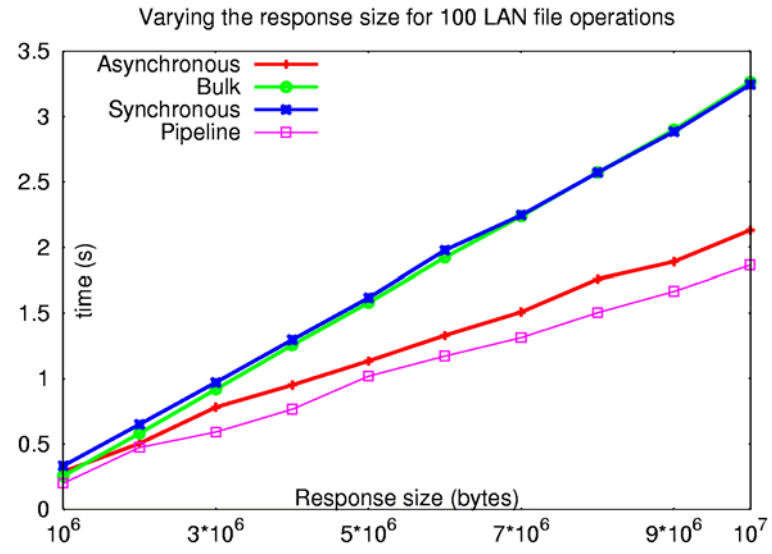
Measuring throughput

- Keeping the number of operations constant (and small) but vary the size of the response
  - Will give an indication of the throughput performance of each model

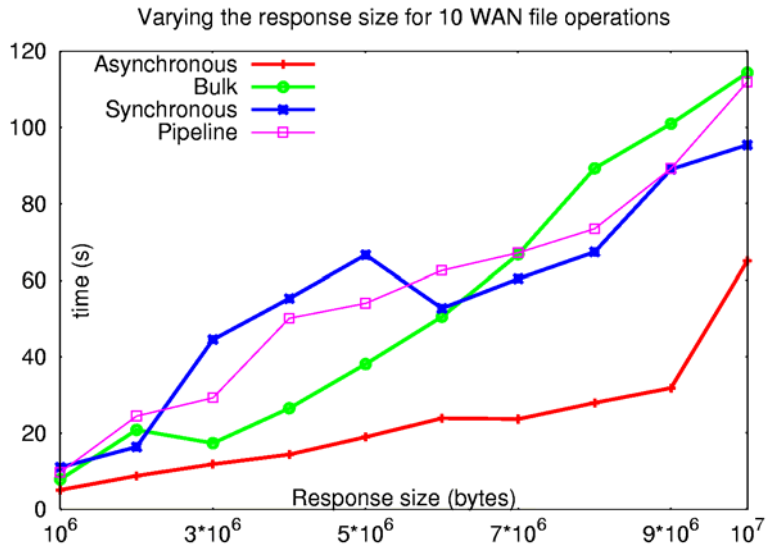
LAN NOOP : async best



LAN FAOP : pipeline advantage



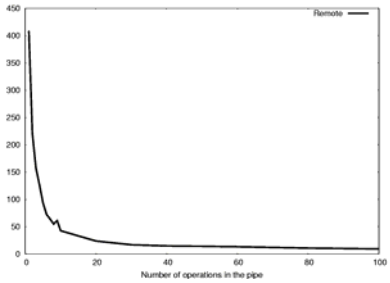
WAN FAOP : transport time dominates



Limiting number of operations

- Limit the number of operations in a bulk while keeping the total number constant, limiting the number of operations in the pipeline

# of bulk ops	LAN	WAN
1	0.25s	3.78s
10	0.25s	5.47s
100	0.24s	15.71s
1,000	0.38s	122.33s
10,000	1.38s	1080.19s
100,000	6.41s	–



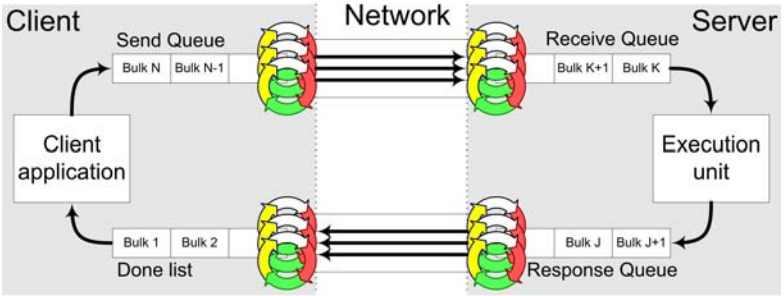
These models do not generally appear like this

- We discussed the “pure” models. However they can be morphed one into the other
- Going from the asynchronous model to the pipeline model

name	threads	connections	queue
general async	n	n	no
pool async	$t < n$	t	no
pool async queue	$t < n$	t	yes
general pipeline	2	2	yes
extended pipeline	$t < n$	2	yes

Combining the models

- Hybrid execution model
  - Configurable number of threads for each segment and number of segments
  - Capacity of executing bulk operations



Conclusions

- Each model has its strength and weakness
- Depending on the exact scenario any model can be the best one
  - Bulk is best for small operations or negligible execution time
  - Pipeline and asynchronous not suitable for many small operations but they gain advantage when execution time (pipeline) or message size (async) increases
  - Performance of async decreases with a large number of operations, bulk and pipeline opposite