Towards Zero Cost I/O:

Met Office Unified Model I/O Server

Martyn Foster

Martyn.Foster@metoffice.gov.uk

A reminder…

Amdahl’s Law

Performance is always constrained by the serial code

$$T_1 = T_s + T_P \qquad T(n) = T_s + \frac{T_P}{n}$$

~~Amdahl's Law~~

The Sales Equation

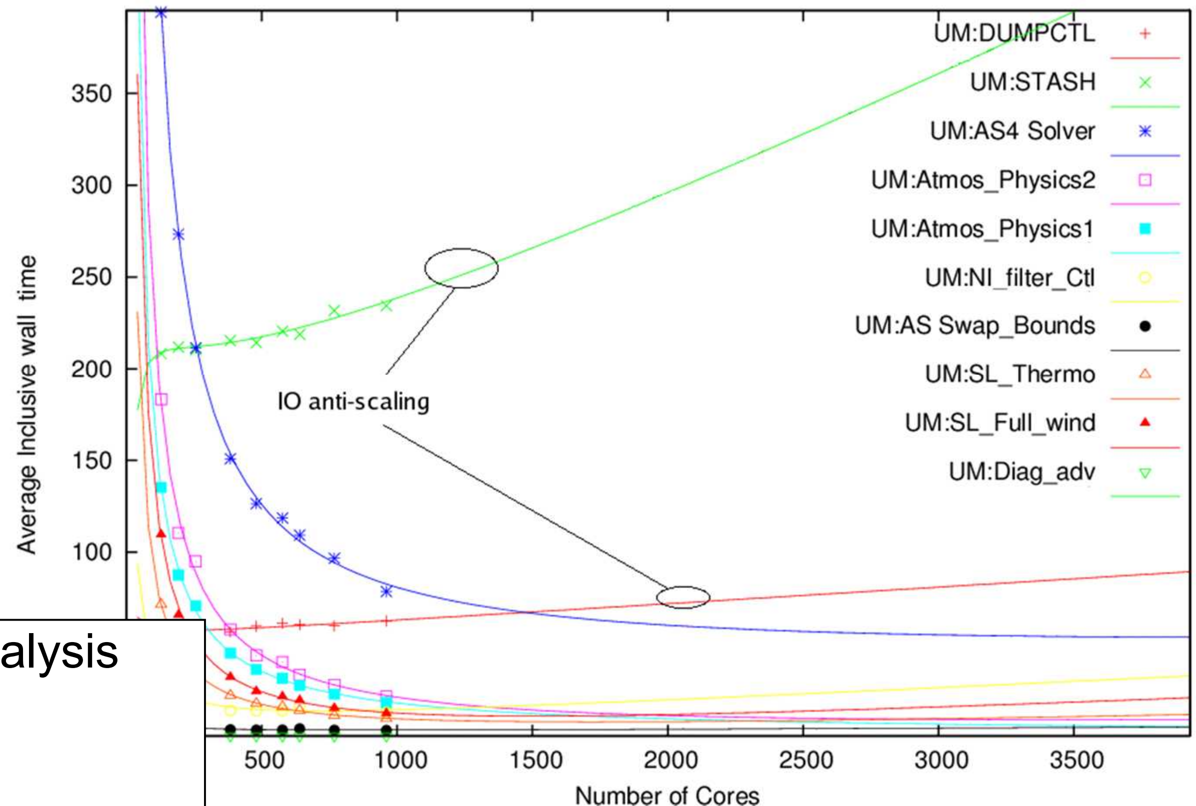
$$T(n) = \sum_{Sections} A + Bn^{-1} + C\ln(n) + Dn + En^{-0.5} + \ldots$$

Gather/scatter

Reductions

limited parallelism

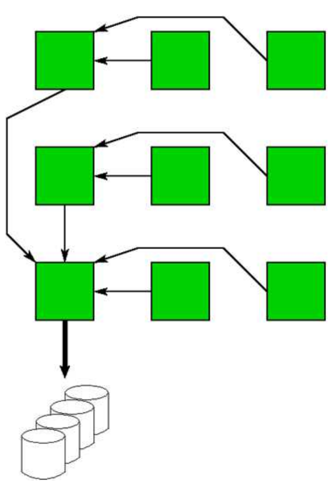
I/O Server Motivation



- Component scaling analysis
  - Version 7.5
  - 5 term fits
- STASH and DUMPCTL
  - Terms in both x and log(x)
  - Will dominate performance at 1500+ tasks.

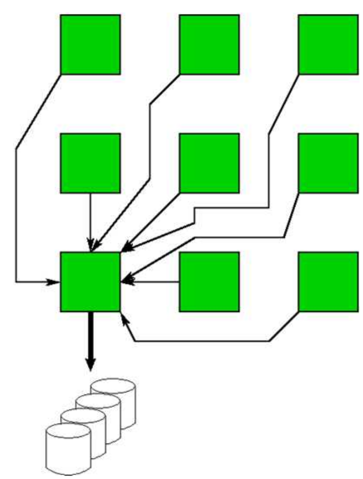
I/O Server Motivation

- Serial time from actual disk I/O
- Antiscaling from gather operations
- Poor Scaling from data packing
    - Sqrt(n) from row wise compression



Traditional diagnostic output

- Gather to row head
- Compress
- Gather to root
- Store



Traditional restart output

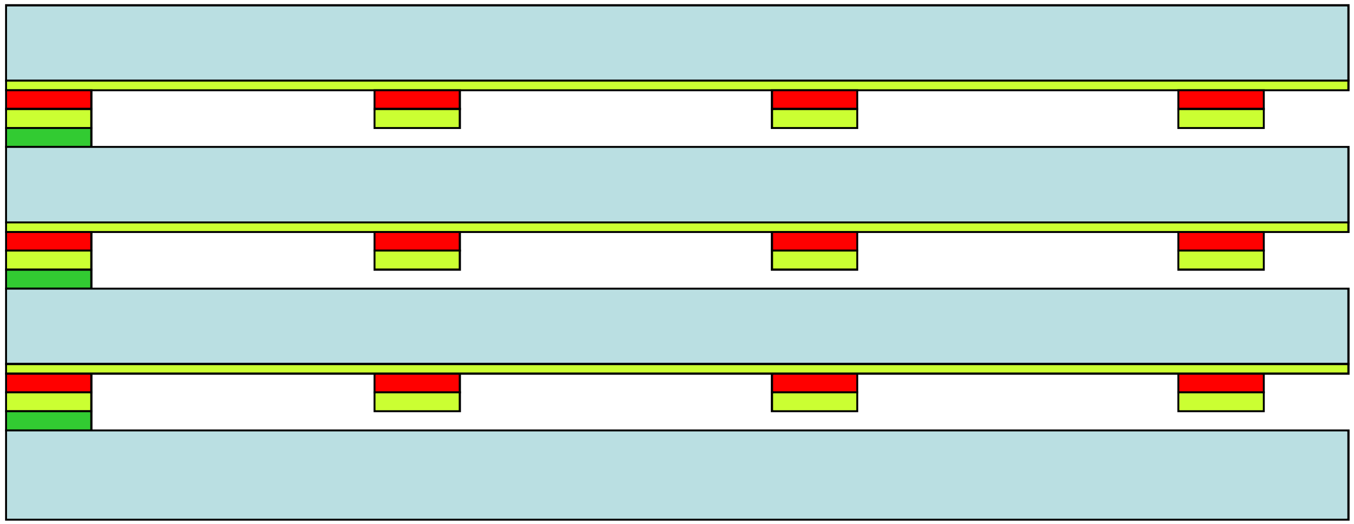
- Gather to root
- Truncate to 32 bit
- Store

Why I/O Server Approach?

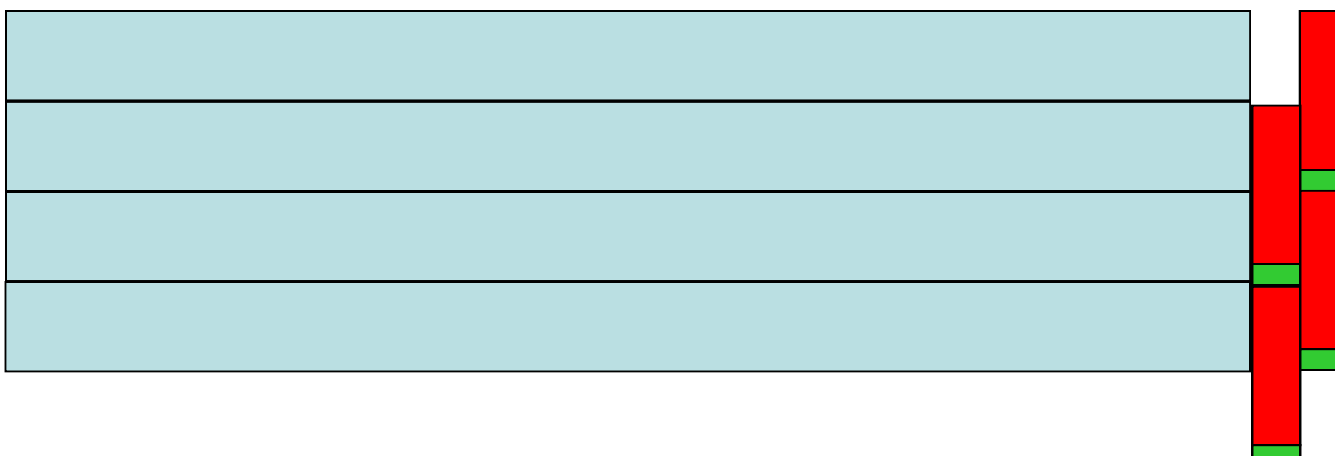
- Full parallel I/O difficult with current packing regime
- Compared to model scalability, CPUs are cheap
- At scale, there is much spare memory available
- Complete offload of packing is possible with an external server
- Exploit the asynchronous nature of data output
  - I/O Should not require synchronisation

I/O Server Motivation

Conventional output for diagnostics
Work
Gather
Pack
Gather
Write

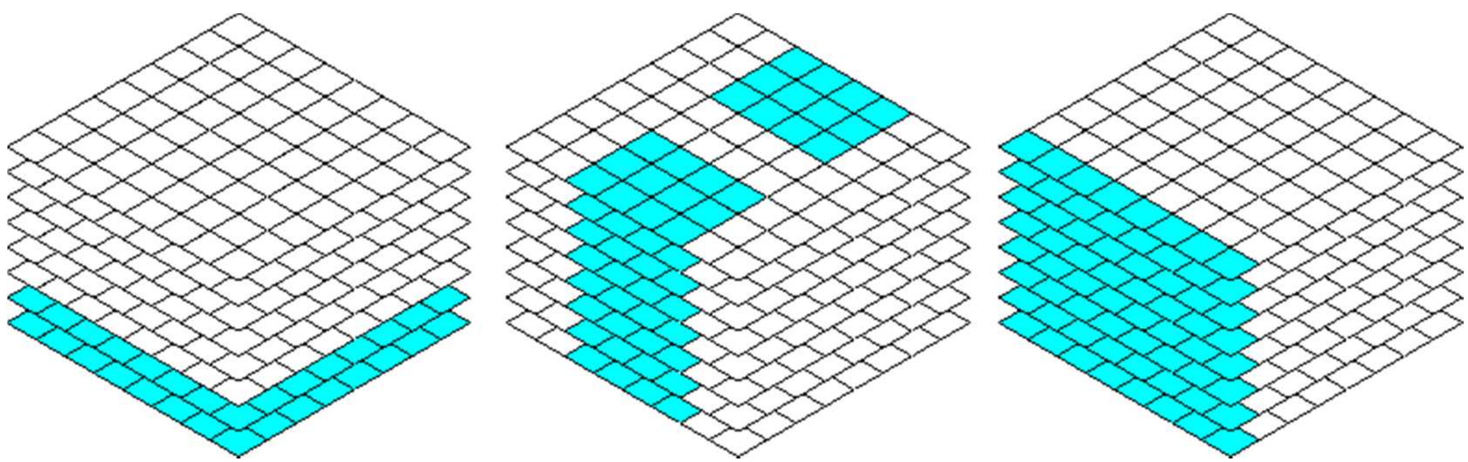


I/O Server approach
Hide commutations
Eliminate dead time



Diagnostic Flexibility

- Various user driven use cases
  - Variables (primary and derived)
  - Temporal processing (e.g. accumulations, extrema, means)
  - Spatial processing (sub-domains, spatial means)
  - Variable to unit mapping
  - Basic output resolution is a 2D field
- Highly variable load with time



I/O Server Design

- I/O proxy server with acceleration
- Basic proxy allows users
  - transparent use of existing (simple) API;
    - Open/close/read/write/getpos/setpos
    - Preserves concept of Fortran units
      - Though, its just used as an opaque handle
  - with some change in detail
    - API calls are collective!
    - Errors not resolvable by the caller
- An MPI pipe to a remote fat buffer
  - Modest runtime savings of ~10%
  - Performance gain is Disk Bandwidth/MPI Bandwidth

I/O Server design

- **Basic proxy design**
  - **Server is threaded**
    - "Listener" receives data & puts in queue
    - "Writer" processes queue including packing
    - Ensures asynchronous behaviour, design goal is to maximise availability of the Listener.
    - Prefers fully multi-threaded MPI implementations, but can work with serialised and funnelled libraries.
- **Shared FIFO queue**
  - Preserves instruction order within an output stream
- **Metadata/Data split**
  - Lightweight control data followed by payload

I/O Server design

- **Acceleration layer is a protocol plugin**.
  - Designed for gather-pack-write operations
  - Triple buffered asynchronous pipeline + write buffer
  - Consume all spare memory!
  - Supports diagnostic (STASH) and restart (DUMP) files, with WGDOS and 32bit compression
    - Extensible to other formats if needed.

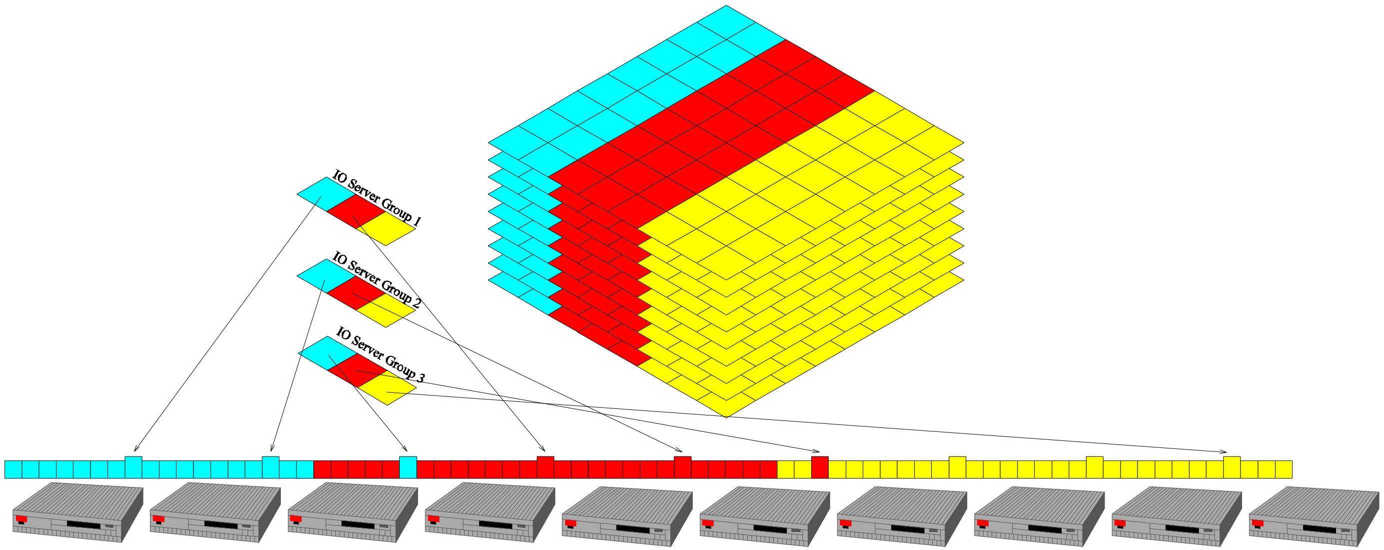
I/O Server design

- Buffer 1: Aggregation over repeated requests to the same unit
  - Evolved from aggregation over levels in a 3D field
  - Trade off: message counts, bandwidth, latency, utilisation
- Buffer 2: Client side dispatch queue
  - Allows multiple in flight transactions with server.
  - Insulates against comms latency and a busy server.
- Buffer 3: I/O Server FIFO
  - Used to facilitate hardware comms offload
- Buffer 4: Aggregation of write data into well formed I/O blocks
  - Filesystem tuneable

I/O Server Design

- Server Parallelism
  - Over units/files
    - Course grain load spreading
    - Units are allowed to 'hop' servers based on current load balance of servers
      - Uses a second 'priority' protocol queue for fast enquiry operations
      - Hints are provided with file_open() calls to signal that future writes to the unit are dependency from prior ones
  - Over North-South domain
    - Independent of model decomposition
    - Allows packing parallelism
    - Multiplies effective server FIFO queue size

I/O Server Deployment

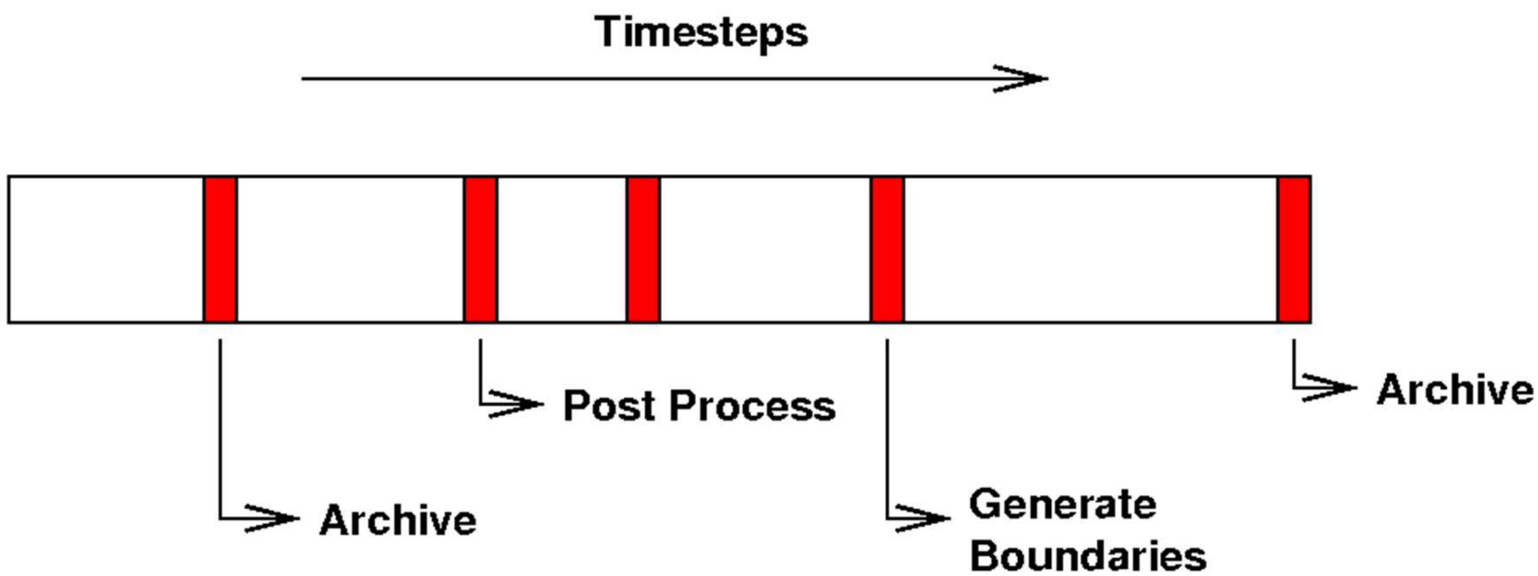


- Server tasks interleaved with simulation tasks
  - Typically one server per node
    - Maximise available node I/O bandwidth
    - Maximise memory use
  - Keep IO server domains in proximity of Atmos domains

Subtlety: Temporal Reliability

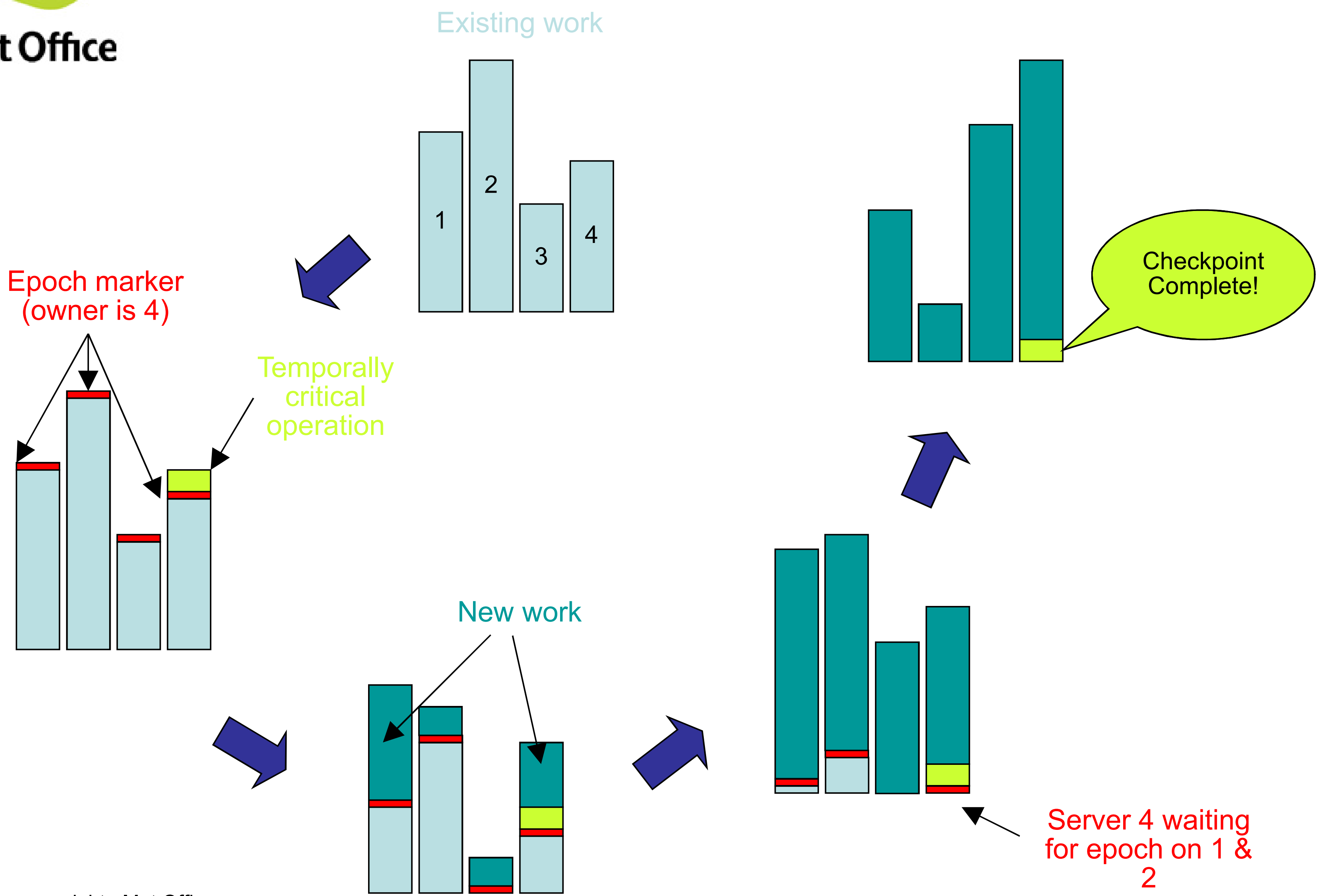
- The operational environment needs to know what's going on
  - But I/O is running in arrears and no occurs in the expected order!
    - Messages to trigger archiving and pre-processing
    - Model state consistency (which is my last known good state?)
- Introduce a new generic ‘epoch’ pseudo operation
  - Server owning the epoch must wait until all other servers have processed the epoch.
  - Provides limited temporal ordering between independent operations where needed
    - at the expense of stalling one server.

Subtlety: Temporal Reliability



- Automatic post-processing
  - Model can trigger automatic post-processing
    - Call outs to monitor processes
  - Requests dealt with by I/O Server
  - FIFO queue + epoch markers ensure integrity of data

Subtlety: Temporal Reliability



Lots of tuneable parameters…

- Number and spacing of I/O servers
  - Core capability and mapping to hardware
- Buffering Choices; Where to consume memory
  - FIFO for I/O servers, Client queue size, Aggregation level
  - Platform characteristics, MPI characteristics, output pattern
- Load balancing options
- Timing tunings
  - Thread back offs, to avoid spin loops on SMT/HT chips
- Standard disk I/O tunings (write block size) etc

Profiling: lock metering, loading logs, transaction logs, comms timers

I/O Server History

	Version	Date
Synchronous proxy server	VN7.7	11/08/10
Asynchronous diagnostics	VN7.8	16/12/10
Asynchronous restart data	VN7.9	27/04/11
Operational rollout (Global)	**PS27**	**July 2011**
Asynchronous metadata protocol, dynamic load balancing	VN8.0	26/08/11
Operational rollout (LAM)	**PS28**	**Jan 2012**
Parallel I/O Servers	VN8.2	30/04/12

Results (PS27)

- 1st Operational configuration
  - 8 serial I/O servers
  - 760 Atmosphere tasks (20x38)
- QU06 Global model (PS27 Code level)
  - 120GB output total
    - 4 x 10.6GB restart files, 25 files total
- Good performance on POWER6
- Migration to POWER7 slows down
  - Atmospheric model faster
  - Exposes previously completely hidden packing time

Config	Performance (relative No-IOS)	
	P575	P775
IO Disabled		42%
No I/O Server	100%	100%
Proxy I/O Server		90%
Asynchronous I/O Server	73%	**154%**

Results (Parallel Servers)

Met Office

- Same QU06 model @24 Nodes on P775.
- Parallel servers remove the packing blockage
- For the same floor space, more CPUs allocated to IO gives better performance
- Moving to 26 Nodes is a super linear speedup.
- Insignificant observable I/O time mid-run, 20 seconds at run termination whilst the final I/O discharges

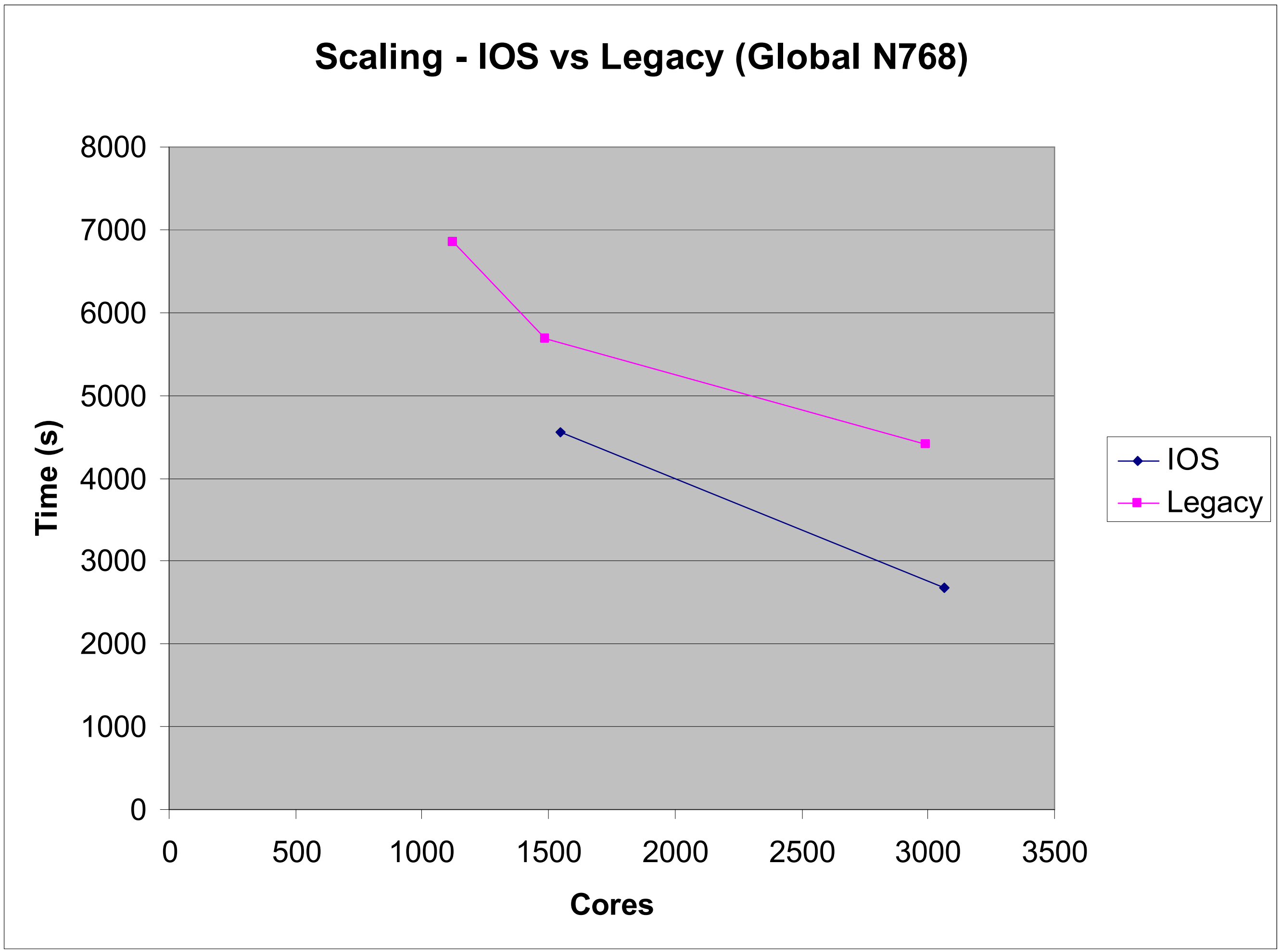
Config	Performance (relative No-IOS)
Original 20x38 atmosphere grid	
8x1	154%
4x2	114%
2x4	77%
1x8	84%
Smaller 22x34 atmosphere grid	
10x2	85%
5x4	58%
4x5	63%
2x10	55%

Results (N768 ENDGAME)

- Potential future global model
  - 96 Nodes, 6144 threads
  - 2.25x more grid points
- Untuned first run!
  - Use 4 servers to reflect rough breakdown of files

Config	Performance (relative No-IOS)
4x1	<timeout>
4x2	130%
4x4	75%
4x8	62%
4x14	60%

Results (N768 ENDGAME)



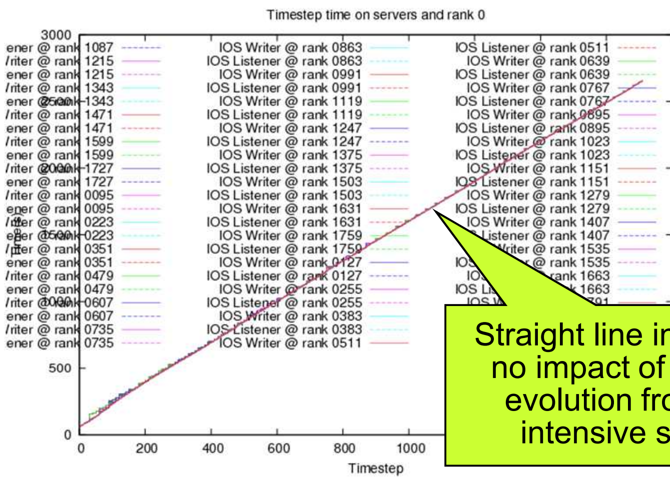
Good/Bad Run Comparison

Met Office

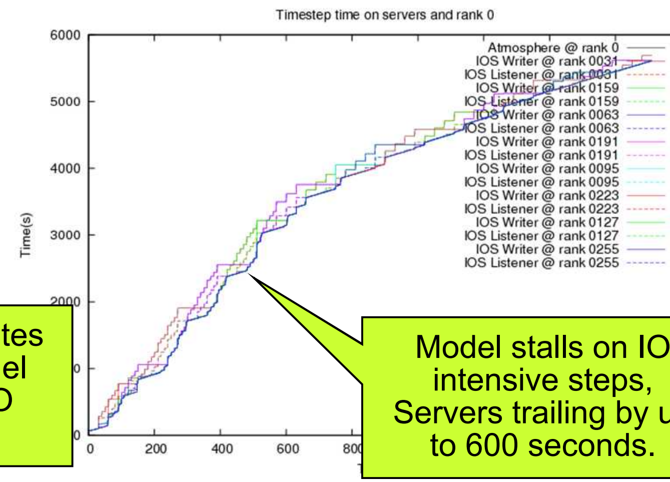
Fast: 4x14

Slow: 4x2

Time vs. Time step

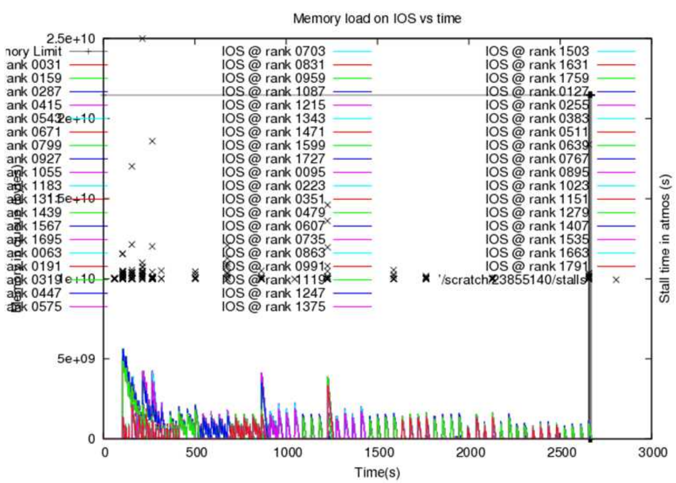


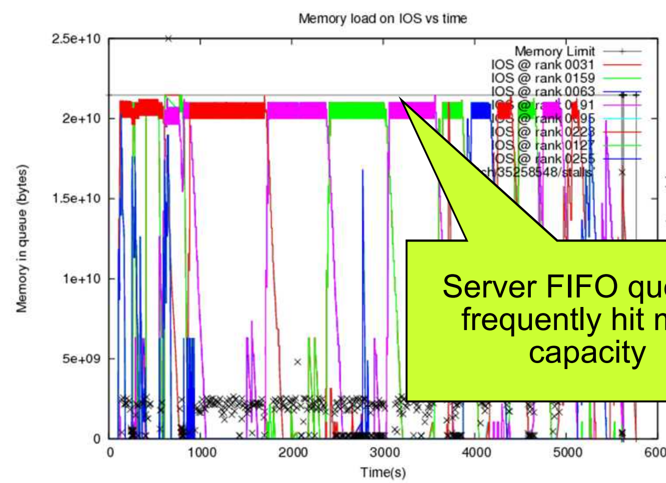
Straight line indicates no impact of model evolution from IO intensive steps



Model stalls on IO intensive steps, Servers trailing by up to 600 seconds.

Backlog vs. Times

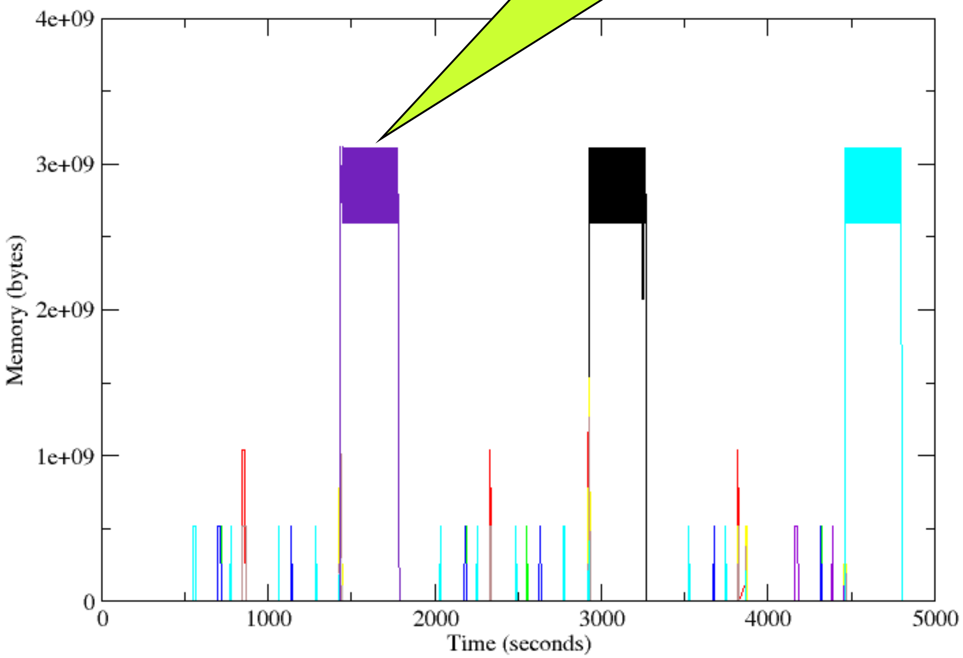




Server FIFO queues frequently hit max capacity

Results High Res Climate

Only a modest server side FIFO queue is used as the nodes do not have memory to buffer a dump – client side buffering is used instead.



- N512 resolution AMIP
- 59 GB restart dumps
- Modest diagnostics
- Cray XE6 with ~9000 cores

- All "in-run" output hidden
- Waits for final restart dump
- Most data buffered on client side

Future work

- Although the servers are parallel, actual I/O still routed through lead process in the group
  - Fail to exploit available disk bandwidth fully
  - Final I/O time doesn't improve with server parallelism
- Implement MPI-IO within an I/O server team
  - Data compression means gathering compressed data sizes is unavoidable.
  - Some deterministically compressed fields can be synchronisation free

Future Work

- I/O server reads are expensive
  - Currently the server is strict FIFO
  - Can't return data until existing backlog is executed.
  - Need to allow out-of-order operation
- Model reads stall execution
  - Boundary files, ancillaries, etc…
  - Implement field prefetch and decode
  - Scatter directly from I/O server
  - Aim to have input data decoded into Server memory before request arrives.

Future Work

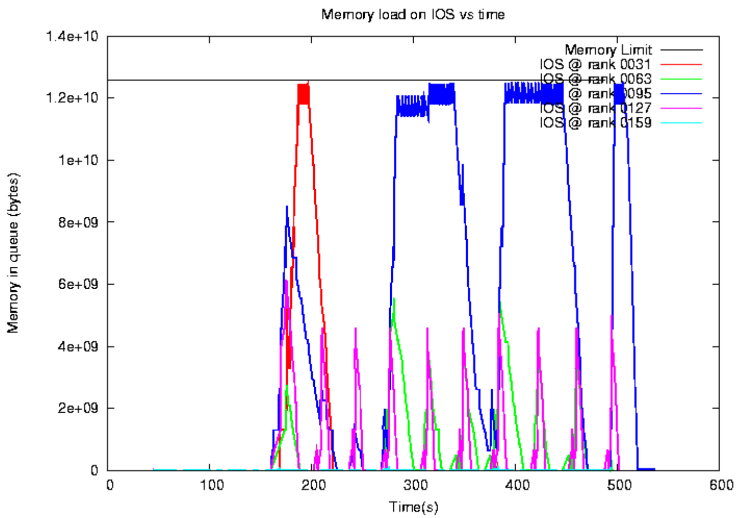
- Valuable features from emerging MPI-3
  - Asynchronous gather/scatter ops should improve on p2p iSend/iRecv method
  - One-sided get/put should reduce protocol overheads
  - Per-communicator MPI tuning
    - Settings for atmos are not ideal for atmos<->I/O server coupling
    - Generically useful in coupled models.

Questions and answers

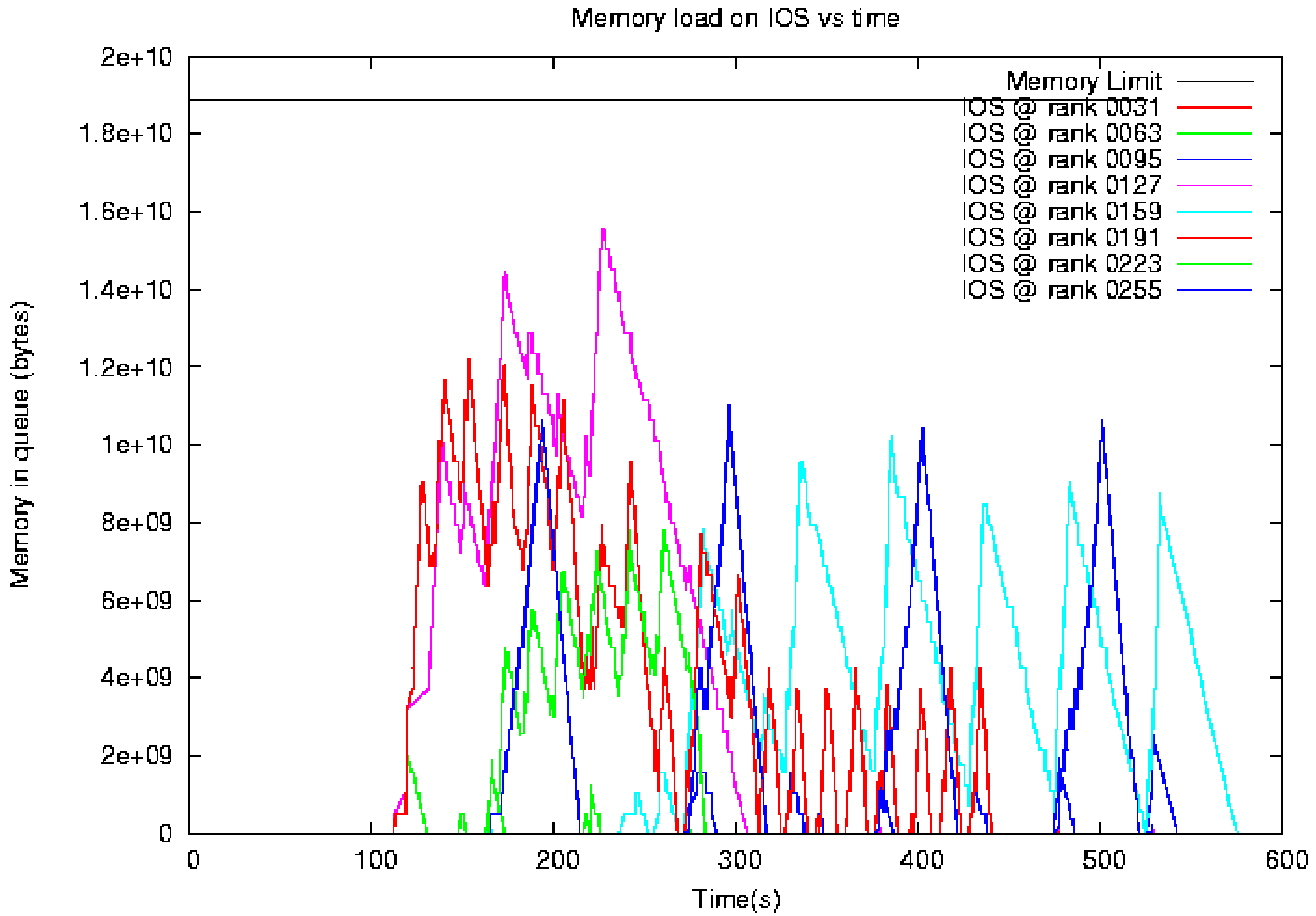
Met Office

Spare slides, you should stop here.

Overloaded servers



I/O Servers keeping up!



MPI considerations

- Differing levels of MPI threading support
  - Best with MPI_THREAD_MULTIPLE
  - OK with MPI_THREAD_FUNNELED

- MPI tuning
  - Want metadata to go as quickly as possible
  - Want data transfer to be truly asynchronous
  - Don’t want to interfere with model comms (e.g. halo exchange)
  - Currently use 19 environment variables!

Deployment

- July 2011 – Operational global forecasts
- January 2012 – Operational LAM forecasts
- February 2012 – High resolution climate work

- Not currently used in
  - Operational ensembles
  - Low resolution climate work
  - Most research work

Global Forecast Improvement

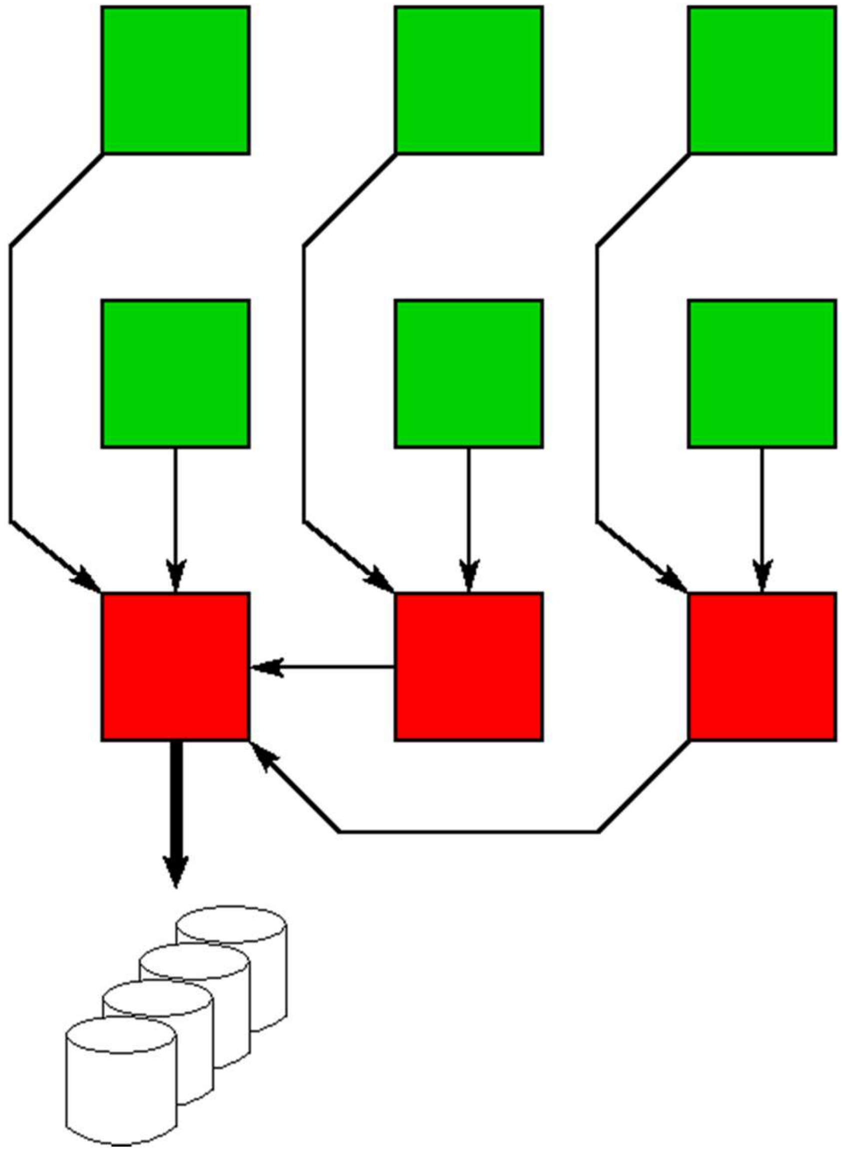
	QG 00/12	QG 06/18	QU
Time	777s	559s	257s
%age	19%	28%	27%

Total saving: over 21 node-hours per day

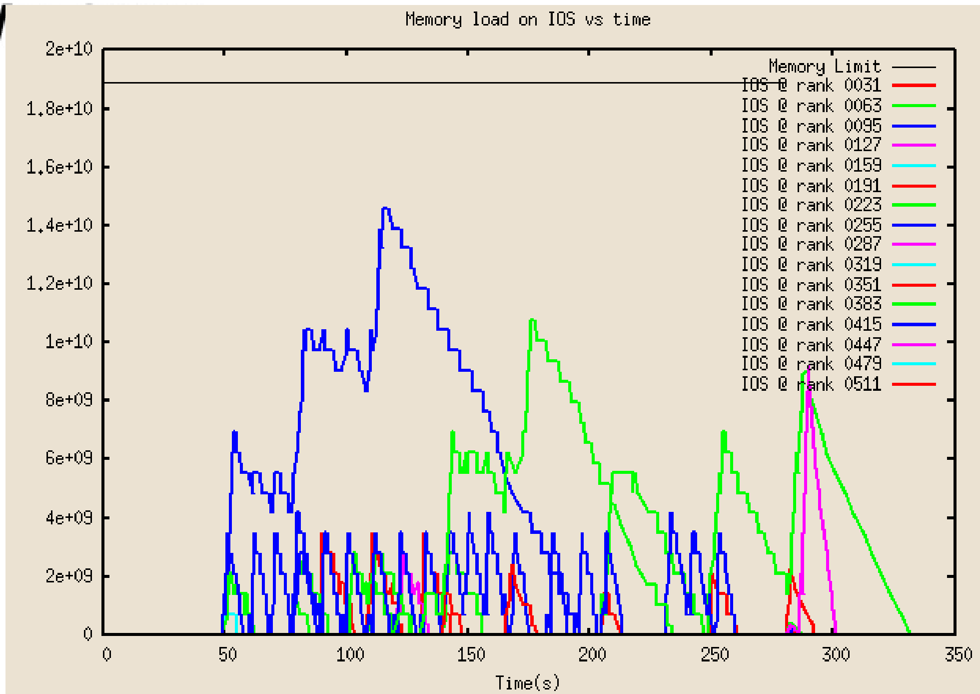
Future Developments

- Parallel I/O
  - Better exploitation of available IO bandwidth
  - Improved shutdown time
- Read ahead
  - Potential for boundary conditions / forcings
  - Some possibilities for initial condition

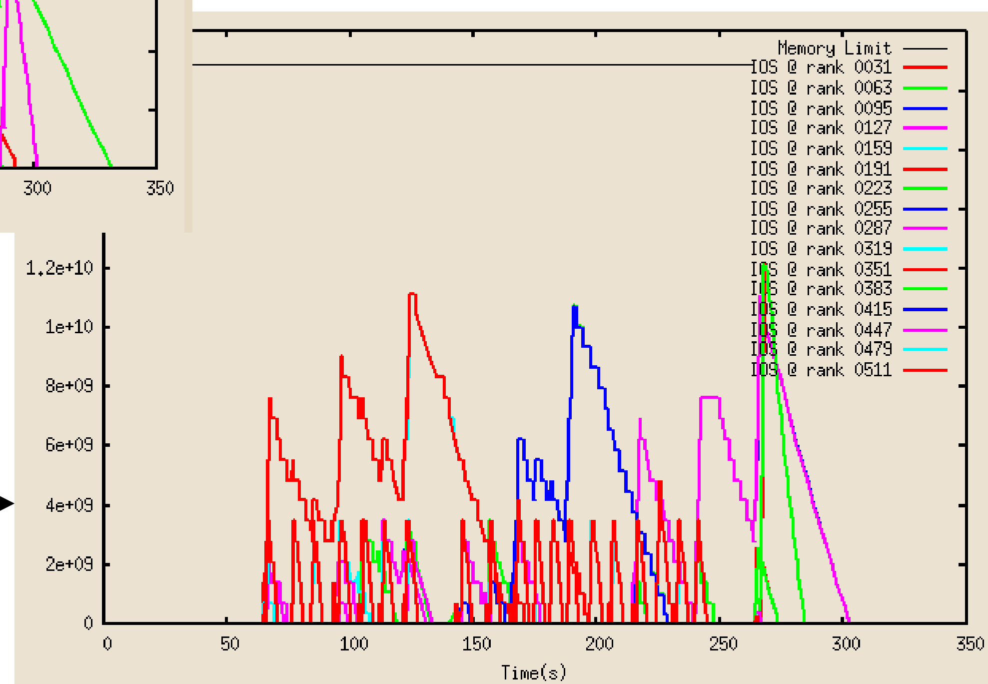
Parallel I/O Servers



Parallel I/O server improvement



Before



After