

# **Global Ocean Prediction with HYCOM**

**Alan J. Wallcraft**

**Naval Research Laboratory**

**1st Petascale Computing and the Geosciences  
Workshop**

**April 4, 2006**

# COMPUTATIONAL ASPECTS OF OCEAN MODELS

- **Typical ocean model is 3-D Finite Difference**
- **Some of the characteristics of a 2-D problem**
- **Vertical scales much different from horizontal**
  - **HYCOM 1/12° fully-global: 4500 x 3298 x 34**
- **2-D domain decomposition for SPMD scalability**
  - **Vertical dimension “on-chip”**
    - **Often treated implicitly**
- **Fast surface gravity waves  $O(100\text{m/s})$** 
  - **$O(100)$ x faster than advective and internal gravity wave speeds**
  - **Separate 2-D sub-problem**
  - **Split-explicit or semi-implicit time step**
- **Static load balance based on land/sea mask**
  - **20% to 40% efficiency gain from skipping land**

# LIMITS ON OCEAN MODEL SCALABILITY

- **2-D sub-problem**
  - **2-D Halo exchanges and 2-D global sums**
  - **Relatively little computational work**
  - **Highly dependent on communication latency**
- **3-D sub-problem**
  - **3-D Halo exchanges**
  - **Still relatively little computational work per halo exchange**
  - **Still dependent on communication latency**
- **I/O**
  - **Typically no overlap between I/O and computations (today)**
  - **Need fast synchronous reads and asynchronous writes**
    - **From system (e.g. MPI-2 I/O)**
    - **At user level (e.g. via “coupler”)**

# PORTABLE LOW LATENCY COMMUNICATIONS

- **If application programmers could target:**
  - **low latency communication hardware**
  - **low latency portable API**
- **This would:**
  - **Reduce the need to “tune” codes**
  - **Allow scaling to more processors**
  - **Expand the range of practical algorithms**
- **At the high end of the HPC market:**
  - **have memory-based low latency hardware**
  - **no portable API to take full advantage of this**
- **Partitioned Global Address Space languages:**
  - **CAF, Co-Array Fortran**
  - **UPC, Unified Parallel C**
  - **Titanium, based on Java**
- **CAF will be in the next Fortran standard**
  - **MPI is so pervasive that we probably need to mix CAF and MPI**
    - **Implementation dependent**

# BIT-FOR-BIT MULTI-CPU REPRODUCIBILITY

- **Repeating a single processor run:**
  - **Produces identical results**
- **Repeating a multi-processor run:**
  - **Produces different results**
    - **Using either OpenMP or MPI**
    - **e.g. fastest global sum is non-reproducible**
  - **Unless programmer explicitly avoids non-reproducible operations**
- **Two levels of reproducibility**
  - **On the same number of processors**
    - **Some scalable libraries provide this**
  - **On any number of processors**
    - **Only “safe” option for code maintenance**
    - **Always requires careful programming**
    - **Can be slower**
    - **Is required for all operational ocean prediction models (e.g. HYCOM)**

# **HYBRID COORDINATE OCEAN MODEL (HYCOM)**

- **Developed from MICOM by a Consortium**
  - **LANL, NRL, U. Miami**
- **Hybrid Vertical Coordinate**
  - **Isopycnal in open, stratified ocean**
  - **Terrain-following in shallow coastal seas**
  - **Z-level in mixed-layer and/or in unstratified seas**
  - **Dynamically smooth transition between coordinate systems via the layered continuity equation**
  - **Isopycnals can intersect bathymetry by allowing zero thickness layers (as in MICOM)**
- **Open Source ocean model**
  - **Greatly increases size of user community**
  - **Result is more capable and better tested model**
  - **<http://ww.hycom.org>**

# OCEAN PREDICTION USING HYCOM

- Both the Navy (NRL and NAVOCEANO) and NOAA (NCEP) have selected HYCOM for their next generation of Ocean Nowcasting and Prediction systems
- See “Ocean Prediction” at <http://ww.hycom.org>
  - NRL has run an  $1/12^\circ$  (7 km) Atlantic testbed weekly since 2003
  - NOAA is operational daily in Atlantic with 4km near-US resolution
- Navy operational system will be  $1/12^\circ$  (7 km mid-latitude) fully global, including a coupled sea-ice model (LANL’s CICE)
  - Ocean array size: 4500 x 3298 x 34
  - Runs on 784 processors (IBM P655+)
  - Per model month:
    - Run time: 21-23 wall hours
    - Daily fields: 525 GB (250 GB compressed)
  - Transitioned from R&D in FY07
- Double resolution to  $1/25^\circ$  by the end of the decade

# DATA HANDLING

- **Data (model output) handling is an often overlooked issue**
  - **Huge datasets**
  - **Moving between compute engine and archive**
  - **Size of long term archive**
- **We try to do as much post-processing as possible as soon as the model run completes**
  - **Before moving data to the archive system**
  - **Different computational needs**
    - **Fewer processors,  
more memory per processor**
  - **Single system with two kinds of nodes, or two systems with a shared filesystem**

# DOMAIN DECOMPOSITION

- **Split the domain into contiguous sub-domains**
  - **Size each sub-domain for equal work and minimal connectivity to other sub-domains**
- **Add a “halo” or “ghost cells” around each sub-domain such that:**
  - **If the halo is up to date:**
    - **Sub-domain operations are independent**
      - **Only using sub-domain and halo values**
- **Domain is distributed across the processors**
  - **Program only has memory for one sub-domain plus its halo**
- **Land can be a large fraction of the total grid**
  - **Primary reason for different domain decomposition strategies in ocean models**
  - **Affects efficiency, not scalability**

## **EQUAL-SIZED RECTANGULAR TILES**

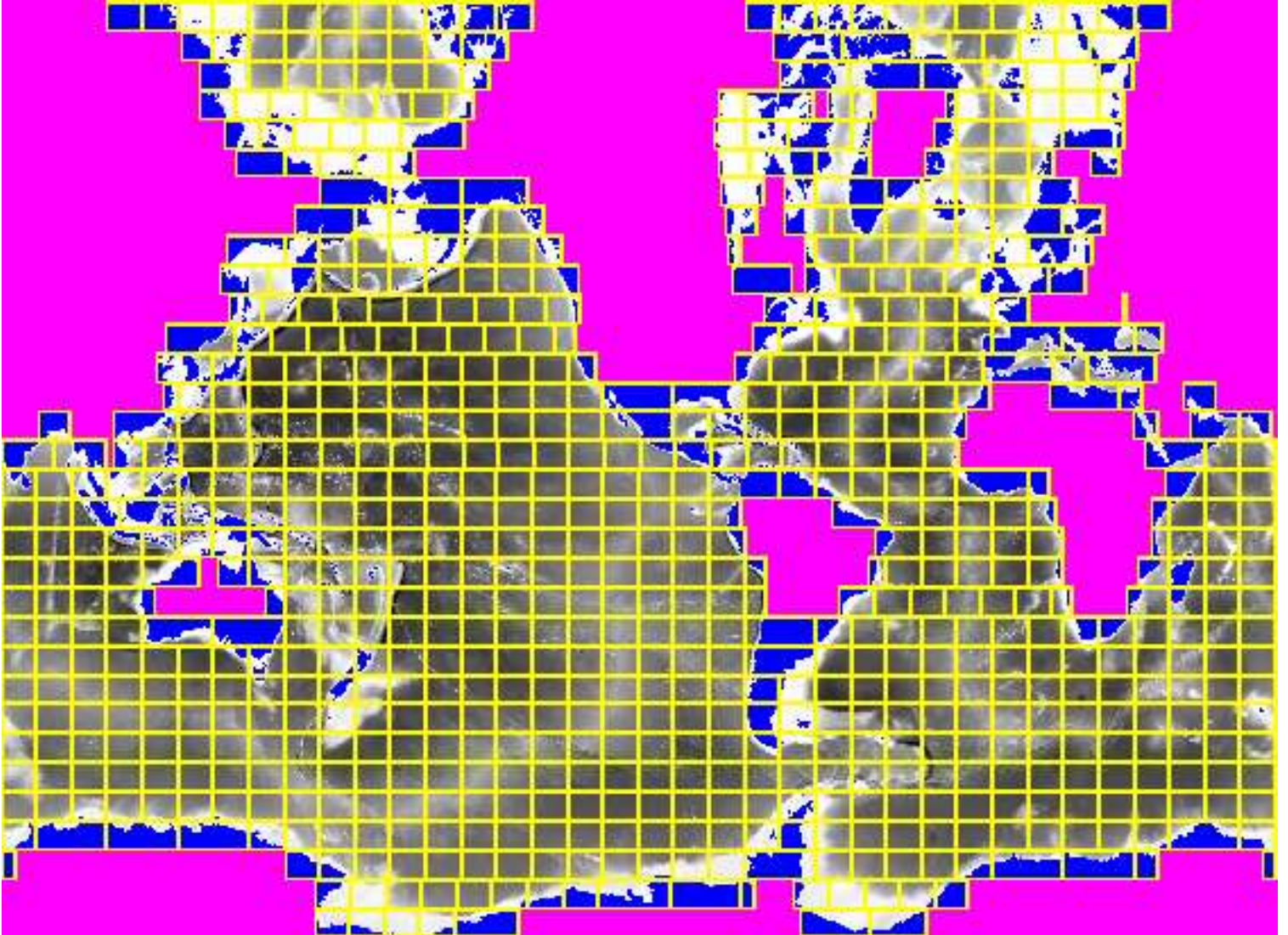
- **Simplest scheme is equal-sized rectangular tiles**
  - **Each tile has four neighbors**
    - **Eight neighbors including halo corners**
- **Overall speed controlled by slowest tile**
  - **Probably have an “all ocean” tile**
    - **no advantage to avoiding land within a tile**
- **So, discard tiles that are entirely over land**
  - **Relatively simple to implement**
  - **Does not discard all land**
  - **Better for large tile counts**
  - **Ineffective on very small tile counts**
  - **MICOM and NLOM**

# **HYCOM'S DOMAIN DECOMPOSITION**

- **Decompose each axis separately**
  - **Still get rectangular tiling**
  - **All tiles in same row are equal height**
  - **Two East-West neighbors**
  - **Many North-South neighbors**
- **Modified equal-area tiling**
  - **Discard all-land tiles**
  - **Shift tiles to fit coastline**
  - **Double-up tiles if less than half ocean**
    - **must avoid land within the tile**
  - **Compared to equal-area tiling:**
    - **Up to 2x the memory requirement**
    - **More expensive halo exchange**
    - **Often significantly fewer tiles**
- **6-element wide halo**
  - **halo is “consumed” over several operations**
  - **reduces the number of communication steps**

## MODIFIED EQUAL AREA TILING

**36x32 = 1152 Tiles but only 781 Active**  
**10% fewer than equal area tiling**



# SCALABILITY TEST

- **Explore scalability to 2,000 processors, of:**
  - **1/12° Global HYCOM (4500x3298x26)**
    - **In DoD TI-0X benchmark suites**
      - **Target of suite is 256 cpus**
    - **A DoD Challenge project configuration**
- **On an IBM P655+ POWER 4+ using the original large global HYCOM TI-05 benchmark, but ignoring the startup time before the first model time step:**

MPI	TASKS	NODES	WALL-TIME	SPEED-UP	
	123	31	5548.3		
	256	52	2740.1	2.02x	123
	504	63	1655.2	1.66x	256
	1006	126	1211.2	1.37x	504

## GLOBAL ON IBM P655+ (I)

- **HYCOM collects subroutine-level timing statistics, which showed that the lack of scalability was due to three operations:**

```
processor  339 out of  504:  
xcsum     time =  211.68351  
xctilr    time =  160.62561  
zaiowr    time =  119.33992  
total     time = 1655.18582
```

```
processor  726 out of 1006:  
xcsum     time =  263.18858  
xctilr    time =  159.35097  
zaiowr    time =  121.35831  
total     time = 1211.19850
```

- **“xcsum” is global sums**
- **“xctilr” is halo exchanges**
- **“zaiowr” is I/O (writes)**

## GLOBAL ON IBM P655+ (II)

- **Benchmark code “frozen” in 2000**
- **Using a recent of HYCOM source code:**

```
processor 450 out of 504:  
xcsum    time = 6.95531  
xctilr   time = 143.62177  
zaiowr   time = 120.44230  
total    time = 1515.25493
```

```
processor 230 out of 1006:  
xcsum    time = 10.05908  
xctilr   time = 125.57702  
zaiowr   time = 122.75476  
total    time = 941.89152
```

- **Time in “xcsum” is much less**
  - **Side-effect of a new advection scheme**
- **Have since improved I/O routines slightly**

## SCALABILITY TEST ON TWO MACHINES (I)

- **On NAVO's kraken (IBM P655+):**
  - **Total I/O time is 88 to 96 seconds**
  - **Without I/O the 1006 to 2040 speedup would be 1.74x**
  - **On 2040 cpus 15% of the time is I/O**

MPI	TASKS	NODES	WALL-TIME	SPEED-UP
	504	63	1515.1	
	1006	126	946.9	1.60x 504
	2040	255	587.2	1.61x1006

- **On ARL's jvn (Linux Networx Xeon Cluster):**
  - **Total I/O time is 284 to 336 seconds**
  - **Without I/O the 1006 to 2040 speedup would be 1.84x**
  - **On 2040 cpus 35% of the time is I/O**

MPI	TASKS	NODES	WALL-TIME	SPEED-UP
	504	252	1867.0	
	1006	503	1209.2	1.54x 504
	2040	1020	772.1	1.57x1006

## **SCALABILITY TEST ON TWO MACHINES (II)**

- **Kraken (IBM) has 8 POWER4+ cpus per node with an IBM switch and GPFS filesystem**
- **JVN (Linux Networx) has 2 Intel Xeon cpus per node with a Myrinet switch and Lustre filesystem**
- **On HYCOM:**
  - **About equal speed for computation and communication**
  - **Kraken is much faster for I/O**
- **It isn't clear if JVN's I/O performance is due to Lustre or to the filesystem hardware**

## SCALABILITY TEST ON CRAY XT3

- On ERDC's sapphire (Cray XT3)
- Slightly different test case, similar I/O needs
  - Total I/O time is 280 to 310 seconds
  - Without I/O the 1006 to 2040 speedup would be 1.97x
  - On 2040 cpus 34% of the time is I/O

MPI TASKS	NODES	WALL-TIME	SPEED-UP
504	504	2321.9	
1006	1006	1403.8	1.65x 504
2040	2040	841.6	1.67x1006

- Lustre filesystem performs similarly on JVN and sapphire

## **HYCOM I/O**

- **Model is REAL\*8, but I/O is big-endian REAL\*4**
- **HYCOM does I/O one 2-D array at a time, from the 1st task only**
  - **Each I/O request is 56.6 MB**
  - **Total I/O is about 11 GB**
  - **Total I/O time of 90 seconds is 125 MB/s**
- **Gather onto 1st task was in REAL\*8**
  - **REAL\*4 gather saved about 20%**
  - **Included in above times**
- **MPI-2 I/O an obvious alternative:**
  - **HYCOM arrays contain “holes” over land**
    - **Must be filled by “data\_void”**
    - **MPI-2 I/O allows gaps, but can’t fill them**
  - **Do (MPI-2) I/O from one task per row**
    - **On both kraken and jvn**
    - **Speeds up reads, but not writes**
      - **HYCOM does far more writes than reads**

## **HYCOM I/O - FUTURE ENHANCEMENTS**

- **Best solution is user-level asynchronous I/O**
  - **Dedicate enough processors to I/O so that all writes can be buffered**
    - **Size of buffer sets number of processors**
  - **Overlap I/O with computation**
    - **Fast I/O still required, since actual I/O time sets lower limit on wall time**
  - **Plan to implement using the Earth System Modeling Framework (ESMF)**

# SUMMARY

- **Low communication latency is one key to good ocean model scalability**
  - **MPI is not a low-latency API**
  - **Co-Array Fortran is a better approach**
- **Bit for bit reproducible global sums are a challenge**
- **I/O is a significant barrier to scalability**
  - **Best solution is user-level asynchronous I/O**
- **Minimize data motion**
  - **Run the model and pre/post processing on:**
    - **Single machine with two kinds of nodes, or**
    - **Two machines with a shared filesystem**