An Introduction to Load Balancing CCSM3 Components

CCSM Workshop

June 23, 2005

Breckenridge, CO

The National Center for Atmospheric Research is funded by the National Science Foundation.

NCAR

Overview

- CCSM3 Introduction
- Load Balance Introduction
- Examples
- Tools vs Log Files

- To learn this material, iteration is required
  - Read, try it, repeat
- This needs to be an interactive session

NCAR

CCSM3 Introduction

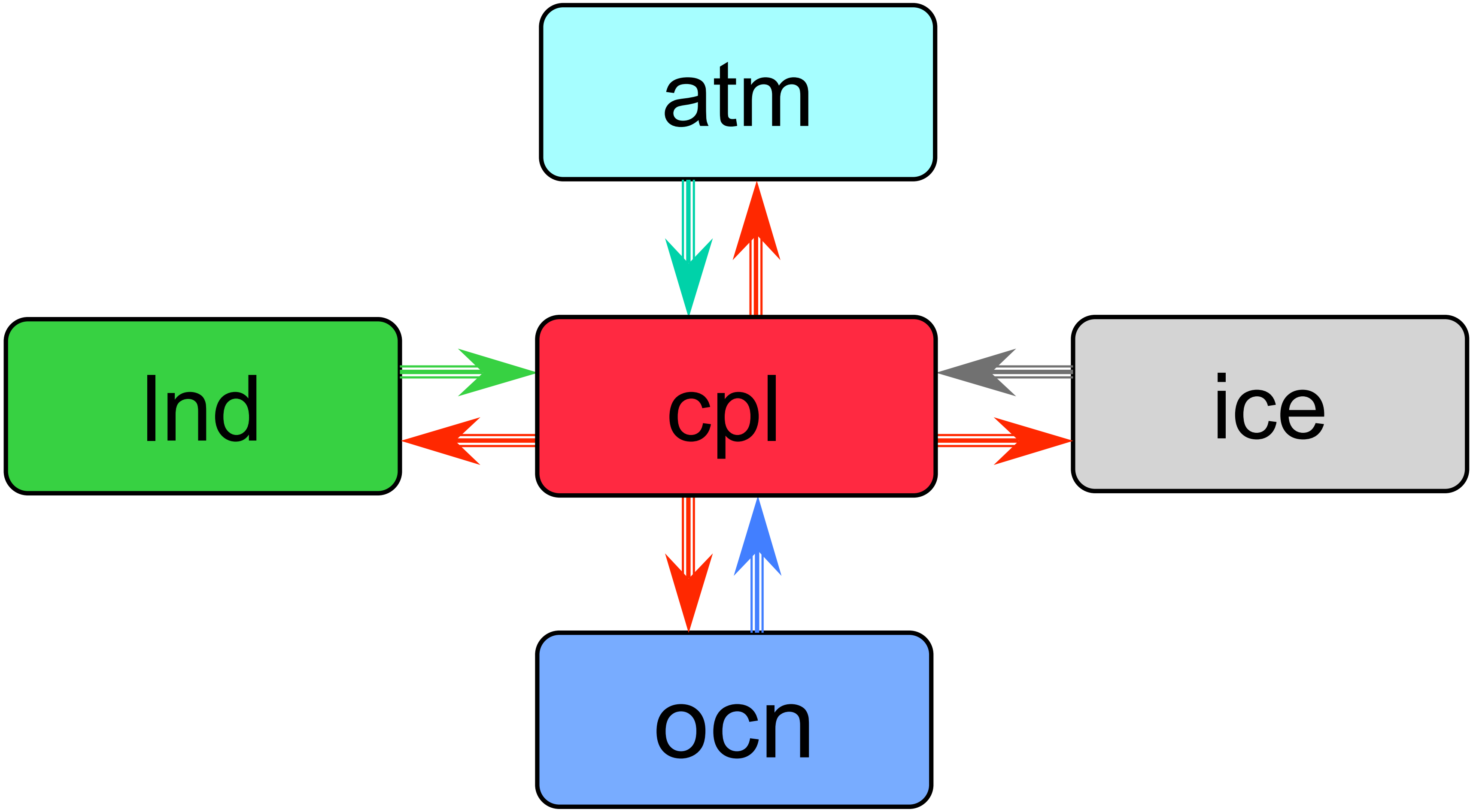
- CCSM, the Community Climate System Model, is a coupled model for simulating the earth's climate system.
  - Developed at NCAR with significant collaborations with DOE, NASA and the university community
- Components in CCSM3 include
  - Atmospheric Model – CAM 3.0

    T31: (48 x 96 x 26) **T42**: (64 x 128 x 26) **T85**: (128 x 256 x 26)
  - Ocean Model – modified version of POP 1.4.3

    3 degree: (100 x 116 x 25) **1 degree**: (320 x 384 x 40)
  - Sea Ice Model – CSIM5 - grid matches ocean
  - Land Model – CLM3 - grid matches atmosphere
  - Coupler - CPL6

NCAR

CCSM Hub and Spoke



NCAR

Performance Metrics

- Raw Performance: Simulated years per wall clock day
  - Capability: Optimize for single job maximum performance
- Performance Efficiency: Simulated years per wall clock day per cpu
  - Capacity: Optimize for system aggregate throughput

NCAR

Two Kinds of "Load Balancing"

- ✓ CCSM load balancing: assigning right number of processors for each component
- o Classic load balancing: moving processing around to even out execution times

NCAR

The CCSM MPMD Balancing Act

- Each component has different scaling attributes in part based on different grid sizes
- System architecture/configuration constraints
  - Node size
  - Queue parameters

NCAR

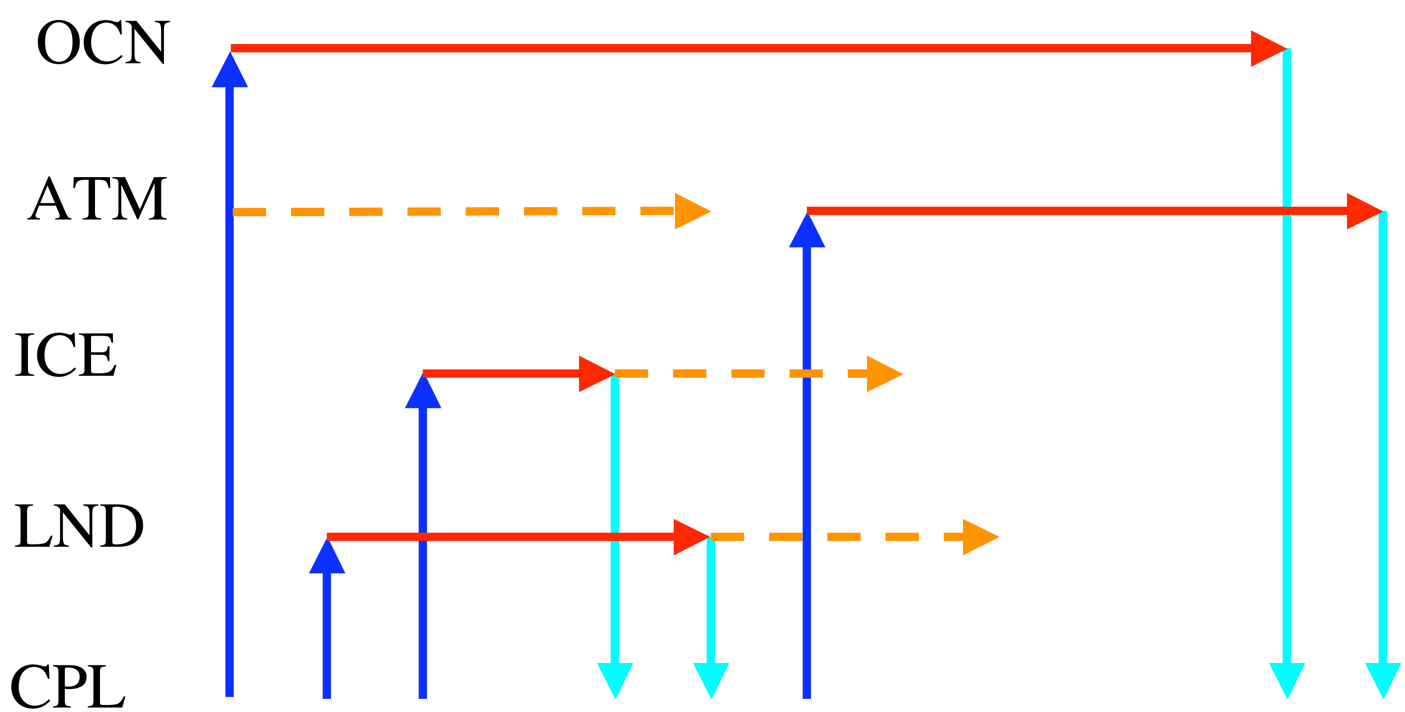
Load Balancing Example

T31x3	OCN	ATM	ICE	LND	CPL	Tot	Yrs/Day
Case 1	4	16	8	8	4	40	20.76
Case 2	2	16	2	8	8	36	22.12

Case 2 used fewer processors and got better performance

NCAR

CCSM3 Process Flow



CPL sending data to component (state 1) [receive]

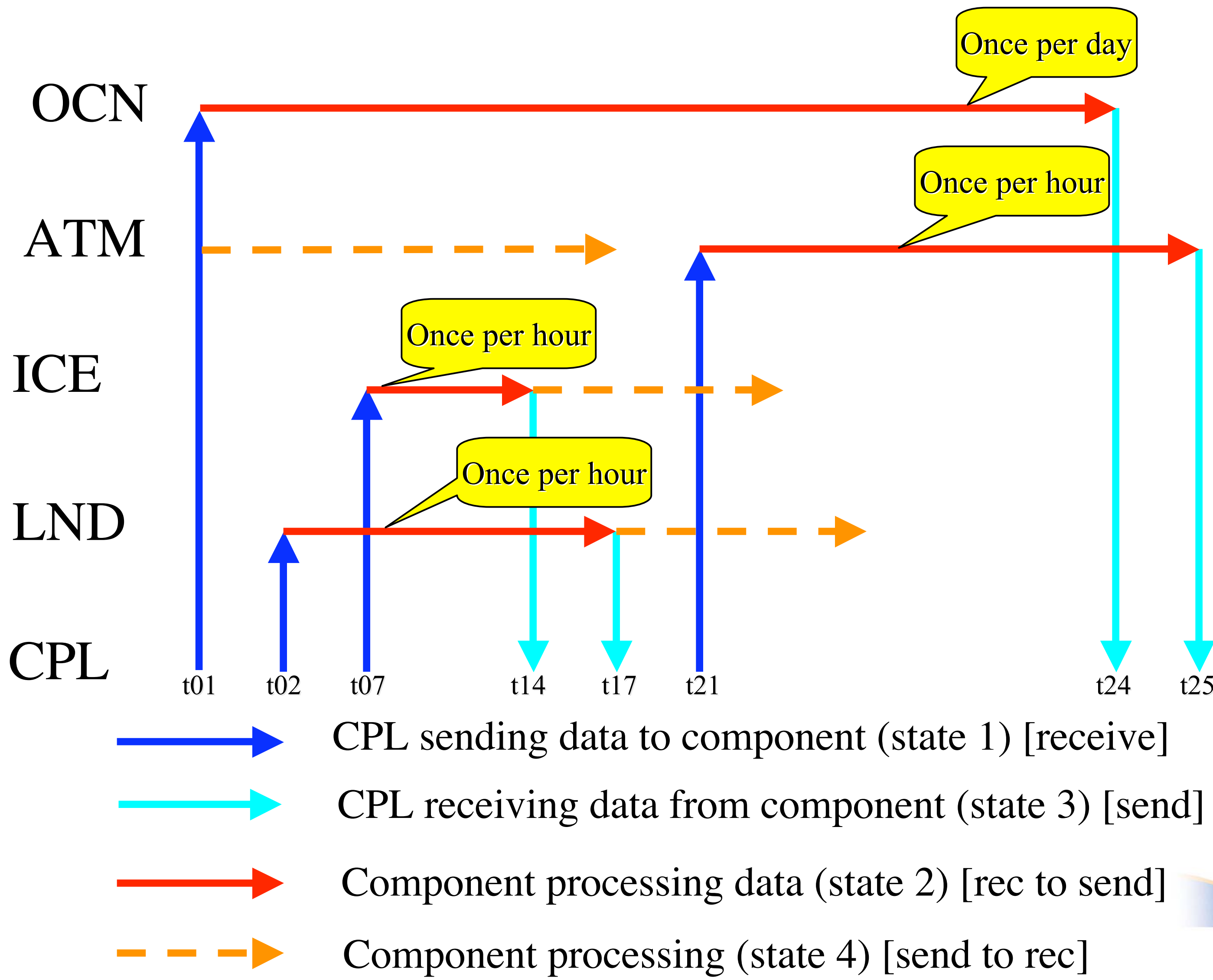
CPL receiving data from component (state 3) [send]

Component processing data (state 2) [rec to send]

Component processing (state 4) [send to rec]

NCAR

CCSM3 Process Flow



NCAR

CPL Log File Timers

```
(shr_timer_print_all) print all timing info:
(shr_timer_print) timer  1:      0 calls,     0.000s, id: ti1 - startup initialization
(shr_timer_print) timer  2:      1 calls,   819.242s, id: t00 - main integration
(shr_timer_print) timer  3:    240 calls,     0.321s, id: t01
(shr_timer_print) timer  4:    240 calls,     2.777s, id: t02
(shr_timer_print) timer  5:    240 calls,     4.692s, id: t03
(shr_timer_print) timer  6:    240 calls,     0.001s, id: t04
(shr_timer_print) timer  7:    240 calls,     1.237s, id: t05
(shr_timer_print) timer  8:    240 calls,     0.175s, id: t06
(shr_timer_print) timer  9:    240 calls,    15.205s, id: t07
(shr_timer_print) timer 10:    240 calls,     0.589s, id: t08
(shr_timer_print) timer 11:    240 calls,     4.596s, id: t09
(shr_timer_print) timer 12:    240 calls,     1.653s, id: t10
(shr_timer_print) timer 13:    240 calls,     4.229s, id: t11
(shr_timer_print) timer 14:    240 calls,     1.899s, id: t12
(shr_timer_print) timer 15:    240 calls,     5.137s, id: t13
(shr_timer_print) timer 16:    240 calls,    63.795s, id: t14
(shr_timer_print) timer 17:    240 calls,     3.649s, id: t15
(shr_timer_print) timer 18:    240 calls,    22.181s, id: t16
(shr_timer_print) timer 19:    240 calls,     8.407s, id: t17
(shr_timer_print) timer 20:    240 calls,     5.114s, id: t18
(shr_timer_print) timer 21:    240 calls,     0.001s, id: t19
(shr_timer_print) timer 22:    240 calls,    16.732s, id: t20
(shr_timer_print) timer 23:    240 calls,     7.187s, id: t21
(shr_timer_print) timer 24:    240 calls,    61.027s, id: t22
(shr_timer_print) timer 25:    240 calls,    16.389s, id: t23
(shr_timer_print) timer 26:    240 calls,     0.263s, id: t24
(shr_timer_print) timer 27:    240 calls,   570.794s, id: t25
```

First

Later

NCAR

CPL Log File "avg dt"

- Can "tail -f" to watch progress of running job

```
(tStamp_write) cpl  model date 0532-04-30 00000s  wall clock 2005-06-22 10:19:00  avg dt   54s  dt   56s
(tStamp_write) cpl  model date 0532-05-01 00000s  wall clock 2005-06-22 10:19:59  avg dt   54s  dt   60s
(tStamp_write) cpl  model date 0532-05-02 00000s  wall clock 2005-06-22 10:20:55  avg dt   54s  dt   56s
(tStamp_write) cpl  model date 0532-05-03 00000s  wall clock 2005-06-22 10:21:50  avg dt   54s  dt   54s
(tStamp_write) cpl  model date 0532-05-04 00000s  wall clock 2005-06-22 10:22:44  avg dt   54s  dt   54s
(tStamp_write) cpl  model date 0532-05-05 00000s  wall clock 2005-06-22 10:23:39  avg dt   54s  dt   55s
(tStamp_write) cpl  model date 0532-05-06 00000s  wall clock 2005-06-22 10:24:35  avg dt   54s  dt   56s
(tStamp_write) cpl  model date 0532-05-07 00000s  wall clock 2005-06-22 10:25:34  avg dt   54s  dt   59s
(tStamp_write) cpl  model date 0532-05-08 00000s  wall clock 2005-06-22 10:26:31  avg dt   54s  dt   57s
(tStamp_write) cpl  model date 0532-05-09 00000s  wall clock 2005-06-22 10:27:26  avg dt   54s  dt   55s
```

- Can see dramatic variation within run
  - Seasonal or longer changes
  - System issues
  - Min, max, mean, mode
  - Can see how fast it should run

NCAR

How Bad Can It Be ("avg dt")?

```
(tStamp_write) cpl  model date 0509-12-05 00000s  wall clock 2004-10-06 17:45:08  avg dt   8s  dt   8s
(tStamp_write) cpl  model date 0509-12-06 00000s  wall clock 2004-10-06 17:45:16  avg dt   8s  dt   8s
(tStamp_write) cpl  model date 0509-12-07 00000s  wall clock 2004-10-06 17:45:28  avg dt   8s  dt  12s
(tStamp_write) cpl  model date 0509-12-08 00000s  wall clock 2004-10-06 17:45:36  avg dt   8s  dt   8s
(tStamp_write) cpl  model date 0509-12-09 00000s  wall clock 2004-10-06 17:45:44  avg dt   8s  dt   8s
(tStamp_write) cpl  model date 0509-12-10 00000s  wall clock 2004-10-06 17:45:52  avg dt   8s  dt   8s
(tStamp_write) cpl  model date 0509-12-11 00000s  wall clock 2004-10-06 17:45:59  avg dt   8s  dt   8s
(tStamp_write) cpl  model date 0509-12-12 00000s  wall clock 2004-10-06 17:46:07  avg dt   8s  dt   8s
(tStamp_write) cpl  model date 0509-12-13 00000s  wall clock 2004-10-06 17:46:15  avg dt   8s  dt   8s
(tStamp_write) cpl  model date 0509-12-14 00000s  wall clock 2004-10-06 17:46:23  avg dt   8s  dt   8s
(tStamp_write) cpl  model date 0509-12-15 00000s  wall clock 2004-10-06 17:46:31  avg dt   8s  dt   8s
(tStamp_write) cpl  model date 0509-12-16 00000s  wall clock 2004-10-06 17:47:13  avg dt   8s  dt  42s
(tStamp_write) cpl  model date 0509-12-17 00000s  wall clock 2004-10-06 17:47:27  avg dt   8s  dt  14s
(tStamp_write) cpl  model date 0509-12-18 00000s  wall clock 2004-10-06 17:47:35  avg dt   8s  dt   8s
```

- 5x impact shown in this case! Can be worse!
- Example: min 10, mode 12, mean 18, max 410

NCAR

CSIM Log File Timers

Timer number 0 Total = 1133.70 seconds min/max = 1133.70 1133.70
Timer number 1 TimeLoop = 827.27 seconds min/max = 827.27 827.27
Timer number 2 Dynamics = 215.45 seconds min/max = 215.45 215.45
Timer number 3 Advectn = 64.28 seconds min/max = 64.28 64.28
Timer number 4 Column = 77.08 seconds min/max = 77.08 77.08
Timer number 5 Thermo = 56.31 seconds min/max = 56.31 56.31
Timer number 6 Ridging = 3.96 seconds min/max = 3.96 3.96
Timer number 7 Cat Conv = 9.75 seconds min/max = 9.75 9.75
Timer number 8 Coupling = 449.50 seconds min/max = 449.50 449.50
Timer number 9 ReadWrit = 4.44 seconds min/max = 4.44 4.44
Timer number 10 Bound = 7.40 seconds min/max = 7.40 7.40
Timer number 11 Pre-cpl = 0.00 seconds min/max = 0.00 0.00
Timer number 12 MPI-send = 15.22 seconds min/max = 15.22 15.22
Timer number 13 MPI-recv = 434.16 seconds min/max = 434.16 434.16
Timer number 14 Snd->Rcv = 323.09 seconds min/max = 323.09 323.09
Timer number 15 Rcv->Snd = 54.79 seconds min/max = 54.79 54.79
Timer number 16 Cpl-recv = 428.28 seconds min/max = 428.28 428.28
Timer number 17 CR-unpck = 2.16 seconds min/max = 2.16 2.16
Timer number 18 CS-pack = 0.97 seconds min/max = 0.97 0.97
Timer number 19 Cpl-send = 12.44 seconds min/max = 12.44 12.44
Timer number 20 = 0.00 seconds min/max = 0.00 0.00

NCAR

POP Log File Timers

Timing information:

```
Timer number  1 Time =      51.92 seconds   EQUATION_OF_STATE
Timer number  2 Time =      41.22 seconds   ANISO
Timer number  3 Time =      41.26 seconds   HMIX_ANISO_MOMENTUM
Timer number  4 Time =      79.87 seconds   HMIX_GM_TRACER
Timer number  5 Time =      77.04 seconds   VMIX_COEFFICIENTS_KPP
Timer number  6 Time =       6.25 seconds   VMIX_EXPLICIT_TRACER
Timer number  7 Time =       0.00 seconds   VMIX_EXPLICIT_MOMENTUM
Timer number  8 Time =      17.69 seconds   VMIX_IMPLICIT_TRACER
Timer number  9 Time =       5.17 seconds   VMIX_IMPLICIT_MOMENTUM
Timer number 10 Time =     290.63 seconds   SEND
Timer number 11 Time =     154.38 seconds   RECV
Timer number 12 Time =     381.75 seconds   RECV to SEND
Timer number 13 Time =       0.00 seconds   SEND to RECV
Timer number 14 Time =      47.31 seconds   ADVECTION_STANDARD_TRACER
Timer number 15 Time =       9.56 seconds   ADVECTION_MOMENTUM
Timer number 16 Time =       0.00 seconds   MOC
Timer number 17 Time =       0.00 seconds   TRACER_TRANSPORTS
Timer number 18 Time =       2.52 seconds   IO_WRITE_TAVG_DUMP_NCDF
Timer number 19 Time =     826.75 seconds   TOTAL
Timer number 20 Time =     821.47 seconds   STEP
Timer number 21 Time =     322.24 seconds   BAROCLINIC
Timer number 22 Time =      34.45 seconds   BAROTROPIC
```

NCAR

CAM Timer Files

Stats for thread 0:

Name	Called	Wallclock	Max	Min
total	1	1134.599	1134.599	1134.599
ccsm_initializa	1	299.906	299.906	299.906
ccsm_rcvtosnd	241	656.489	10.183	2.229
ccsm_runtotal	1	827.451	827.451	827.451
stepon	1	827.451	827.451	827.451
stepon_startup	1	0.015	0.015	0.015
radcswmx	8640	377.275	0.060	0.033
radclwmx	8640	105.468	0.138	0.001
ccsm_snd	240	1.991	0.045	0.003
ccsm_sndtorcv	240	133.871	1.984	0.000
ccsm_rcv	240	35.091	2.526	0.002
ac_physics	481	15.561	0.042	0.031

NCAR

CLM Timer Files

Stats for thread 0:

Name	Called	Wallclock	Max	Min
lnd_timeloop	1	827.755	827.755	827.755
clm_driver	482	825.743	8.654	0.245
lnd_recv	241	663.383	8.378	2.179
lnd_recvsend	240	124.379	1.345	0.426
loop1	481	106.095	0.353	0.177
drvinit	481	0.524	0.002	0.001
clm_driver_io	481	2.001	1.970	0.000
wrapup	481	0.019	0.000	0.000
surfalb	240	2.472	0.015	0.007
lnd_send	240	15.415	0.186	0.026
lnd_sendrecv	240	24.562	2.105	0.063
rtm_calc	80	2.991	0.046	0.033
rtm_update	80	0.379	0.006	0.004
rtm_global	80	1.440	0.024	0.015

NCAR

The getTiming.csh script

- Does not work with all component options (ex. DATM). Will need to look at log files.
- Assumes LOGDIR set to "" (no log dir)
- Assumes short term archive turned off
- Assumes a fully qualified path is given to the tdir parameter (note that "." will not work)
- cd ${CASEROOT}; ${CCSMROOT}/scripts/ccsm_utils/Tools/timing/getTiming.csh -mach <machine name> -tdir `pwd`

NCAR

getTiming.csh Table

Note: for cpl, send=t3~t6, recv=t8~t13, s-r=t15~t16, r-s=t18~t20

p0	atm	lnd	ice	ocn	cpl
conf	8*1	1*1	1*1	1*1	1*1
total	827.451	827.755	827.27	826.75	819.242
send	0.900	0.186	12.44	290.63	6.105
recv	27.981	8.378	428.28	154.38	18.103
s-r	510.633	2.105	323.09	0.00	25.83
r-s	656.489	1.345	54.79	381.75	21.847

STOP_N is 10. simulationyears/day): 2.88

----

s-r/r-s/(sum of s-r and r-s) (for cpl send/recv/s-r/r-s)

cpus	atm	lnd	ice	ocn	cpl
1	-	0.2/0.1/0.3	32.3/5.4/37.7	0/38.1/38.1	0.6/1.8/2.5/2.1
8	51/65.6/116.6	-	-	-	-

NCAR

Script cplstats

```
#! /bin/csh

alias MATH 'set \!:1 = `echo "\!:3-$" | bc -l`'
tail -100 cpl > cpls
set val = `grep "id: t01" cpls | cut -b43-52`
echo "t01 = $val"

set val = `grep "id: t02" cpls | cut -b43-52`
echo "t02 = $val"

set val3 = `grep "id: t03" cpls | cut -b43-52`
set val4 = `grep "id: t04" cpls | cut -b43-52`
set val5 = `grep "id: t05" cpls | cut -b43-52`
set val6 = `grep "id: t06" cpls | cut -b43-52`
MATH val = $val3 + $val4 + $val5 + $val6
echo "t3-6 = $val = $val3 + $val4 + $val5 + $val6"

set val = `grep "id: t07" cpls | cut -b43-52`
echo "t07 = $val"

… etc. …
```

NCAR

cplstats Output Example

```
>> cplstats
t01 = 0.321
t02 = 2.777
t3-6 = 6.105 = 4.692 + 0.001 + 1.237 + 0.175
t07 = 15.205
t8-13 = 18.103 = 0.589 + 4.596 + 1.653 + 4.229 + 1.899 + 5.137
t14 = 63.795
t15-16 = 25.830 = 3.649 + 22.181
t17 = 8.407
t18-20 = 21.847 = 5.114 + 0.001 + 16.732
t21 = 7.187
t22-23 = 77.416 = 61.027 + 16.389
t24 = 0.263
t25 = 570.794
```

NCAR

Walkabout

- CASEROOT
- EXEROOT
- CCSM web
- CSEG web
- CSEG web (internal)
- Bulletin Board
- Log file examples
- Hard copy: spreadsheet, charts

NCAR

What Times Are Looked At?

- Timers do not add up
  - Different binaries measuring somewhat different things
  - Aggregation of timer issues
- When min? When max?
  - Transfer times use minimum
  - “Computational” times use maximum
  - Sanity checks to look at spread and variance
- Variation in timers
  - Load imbalance
  - Seasonal and longer variation
  - System events

NCAR

What To Do? - Ground Rules

- Some hard limitations - cannot use completely arbitrary numbers of processors
- Start from a previously useful scenario
- Choose wisely (luck is ok)
  - Some problems identified at compile, some at runtime
  - Your exploration may lead you to options that may not be obvious … try them
    - Ex. on IBM bluesky, using 20x4 = 80 CPUs
    - Ex. on IBM thunder, using 6x8 = 48 CPUs

NCAR

Ground Rules (cont.)

- Keep records (paper, web, spreadsheets)
- Errors come out at various places (some at build time, some run time)
- 10 day run is only an estimate which may be impacted by
  - Seasonal variations
  - Annual variations
  - Longer term variations
  - Current timers do not make looking at these issues easy

NCAR

Component Set Issues

- Unless otherwise stated all examples are fully coupled (i.e. Component set B with POP, CAM, CSIM, CLM, and CPL)
- General process applies to other choices

NCAR

Data Decomposition Observations

- CAM
  - Must be factor of 2
  - May be factor of 3 or 5
  - Maximum MPI tasks based on resolution (T31 - 48, T42 - 64, T85 - 128)
  - Might be good to be an integral factor of max resolution size
  - Often good to fit into node reasonably
  - Might be able to use MPI and OpenMP
  - Minimum of 2 (all others have minimum of 1)
  - Number of processors does not change the numeric results (not true of all)
- CPL
  - More flexible (can use odd prime numbers for example)
  - “Good” integral factors still seem to be better
- Others
  - Similar kinds of decomposition guidelines

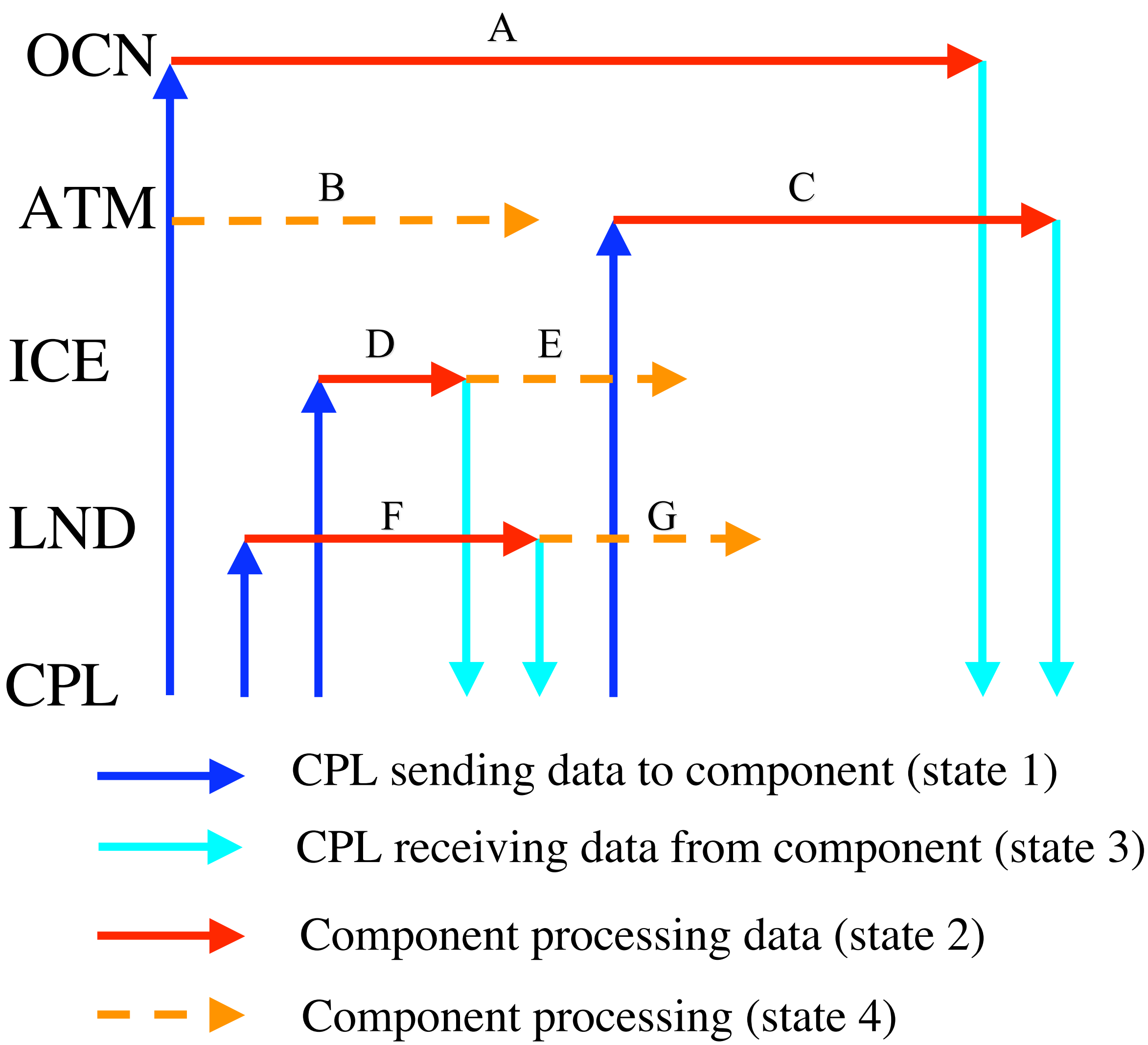
NCAR

Some Additional Items

- Data model only run on 1 CPU
- Group like processes to same node to fill nodes and reduce communication
  - Rearrange COMPONENTS in env_mach.<machid>
    - set COMPONENTS = ($COMP_CPL $COMP_ICE $COMP_LND $COMP_OCN $COMP_ATM)
- You can't always "balance" the model
- I/O can be very important (including LOG files) including your neighbor's use of it
- Your neighbor's use of the network can be very important even if you can't control it
- Where your nodes are on the network can be very important even if you can't control it
- Reducing runtime of one component can improve another particularly when on same node
- Things will change

NCAR

CCSM3 Process Flow



- Targets
  - A <= B+C
  - D < B
  - F < B
  - G < C
  - E < C
  - D < F
- Observations
  - B < C
  - D < E
  - F > G
  - Scaling of B different than C
  - CPL/ICE/LND will allows have idle time

NCAR

OK … How To Go About It?

- Start with CAM
  - Majority of CPUs assigned to CAM
  - Look at integral factors of resolution
  - Look at node size factors
  - Consider OpenMP option (where possible)
- Match POP to CAM processing time
- Pick smallest reasonable number of CPUs for other components such that CAM is not delayed

NCAR

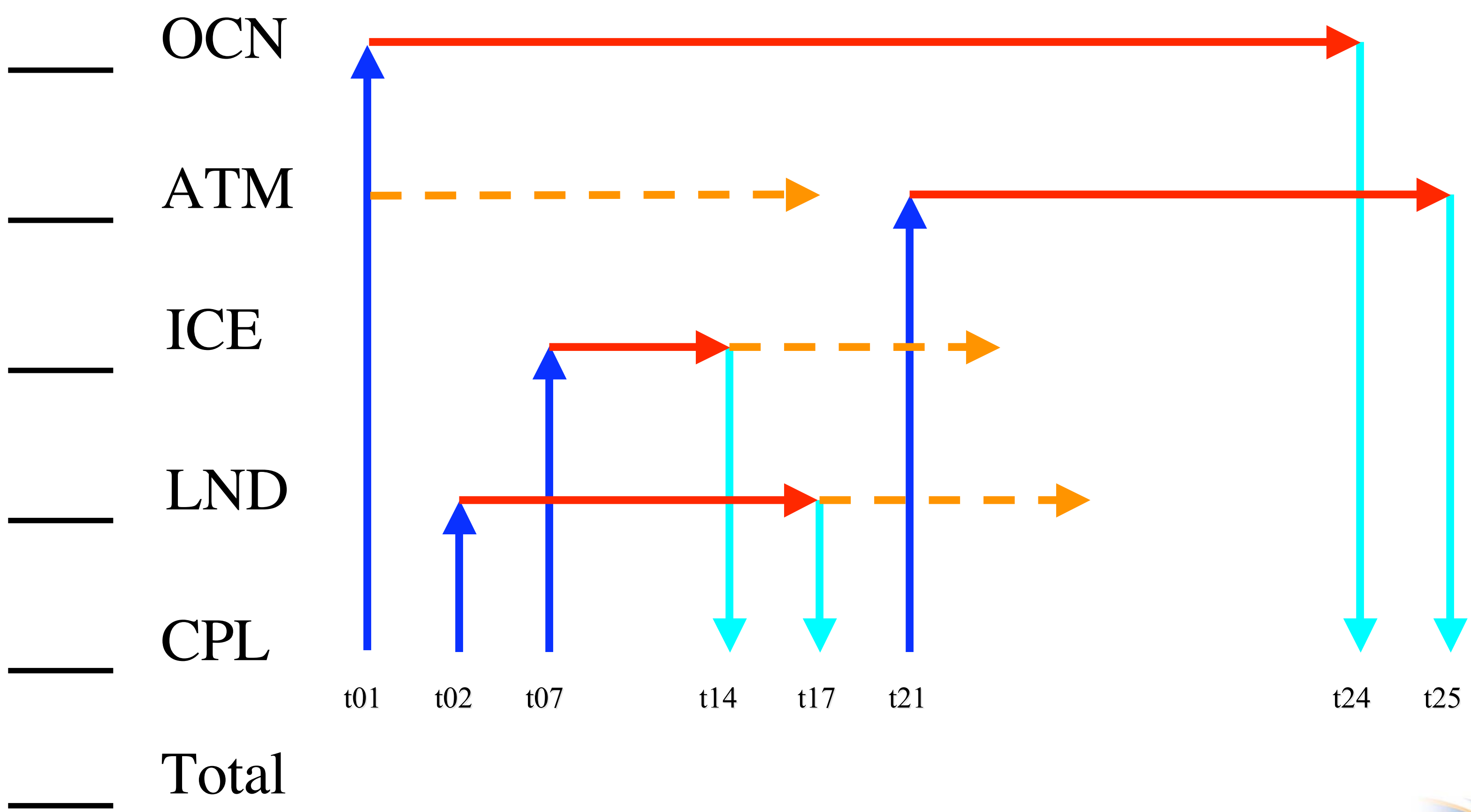
OK … What Do I Really Do?

- Pick a configuration … try it
- Look at CAM
  - Are there MPI wait times?
  - Which? Why?
- Compare POP to idealized CAM time
- Look at ICE and LND
  - Compare "compute" time phases to CAM
  - Examine MPI wait times
- Look at CPL times
  - "Compute" phases
  - Transfer phases
- Change a couple things and try it

NCAR

CCSM Version ____________
Machine ____________
Date ____________
Resolution ____________
Config # ____________

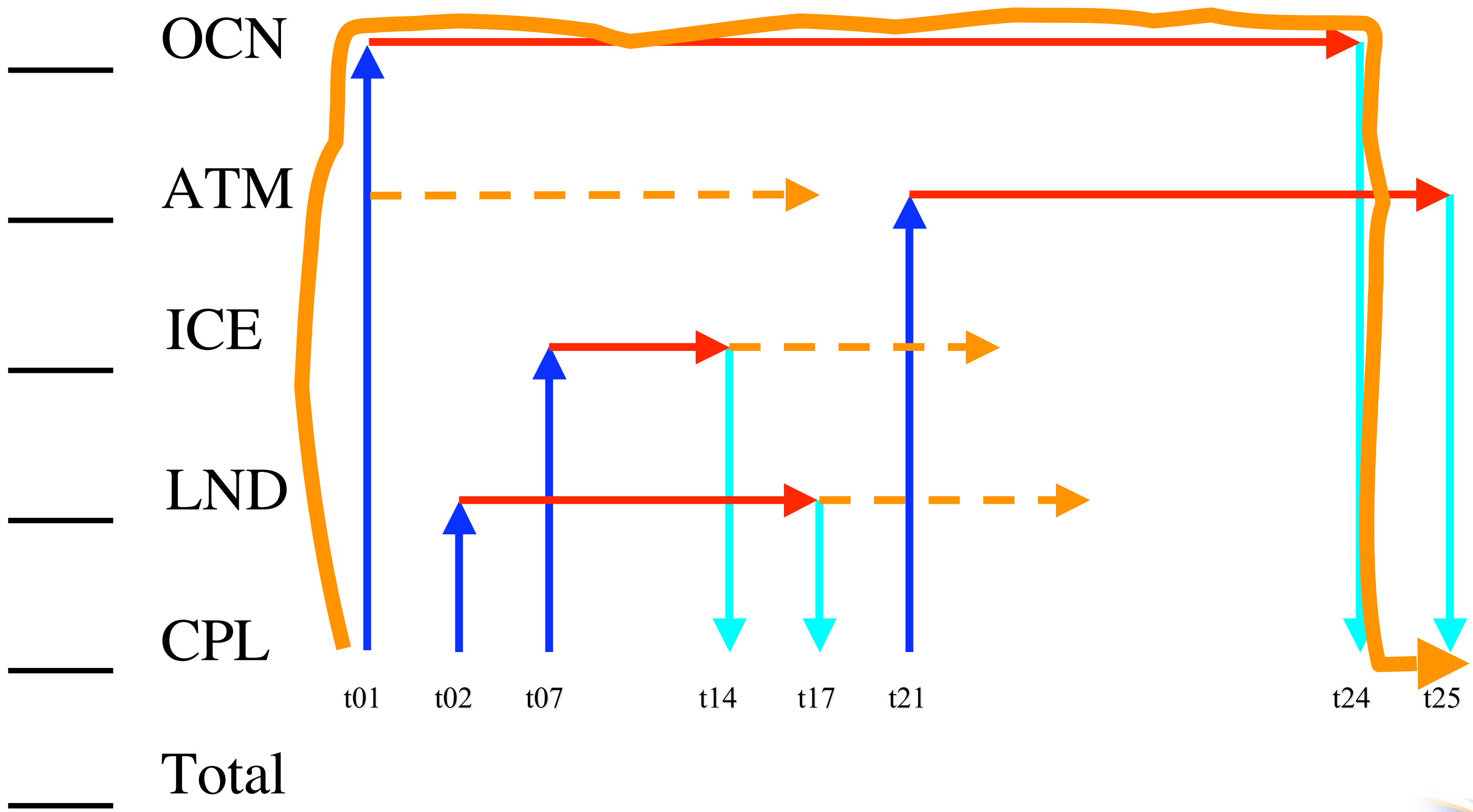
Years/day ____________
Years/day/cpu ____________
CPL main time ____________
CPL avg dt ____________



NCAR

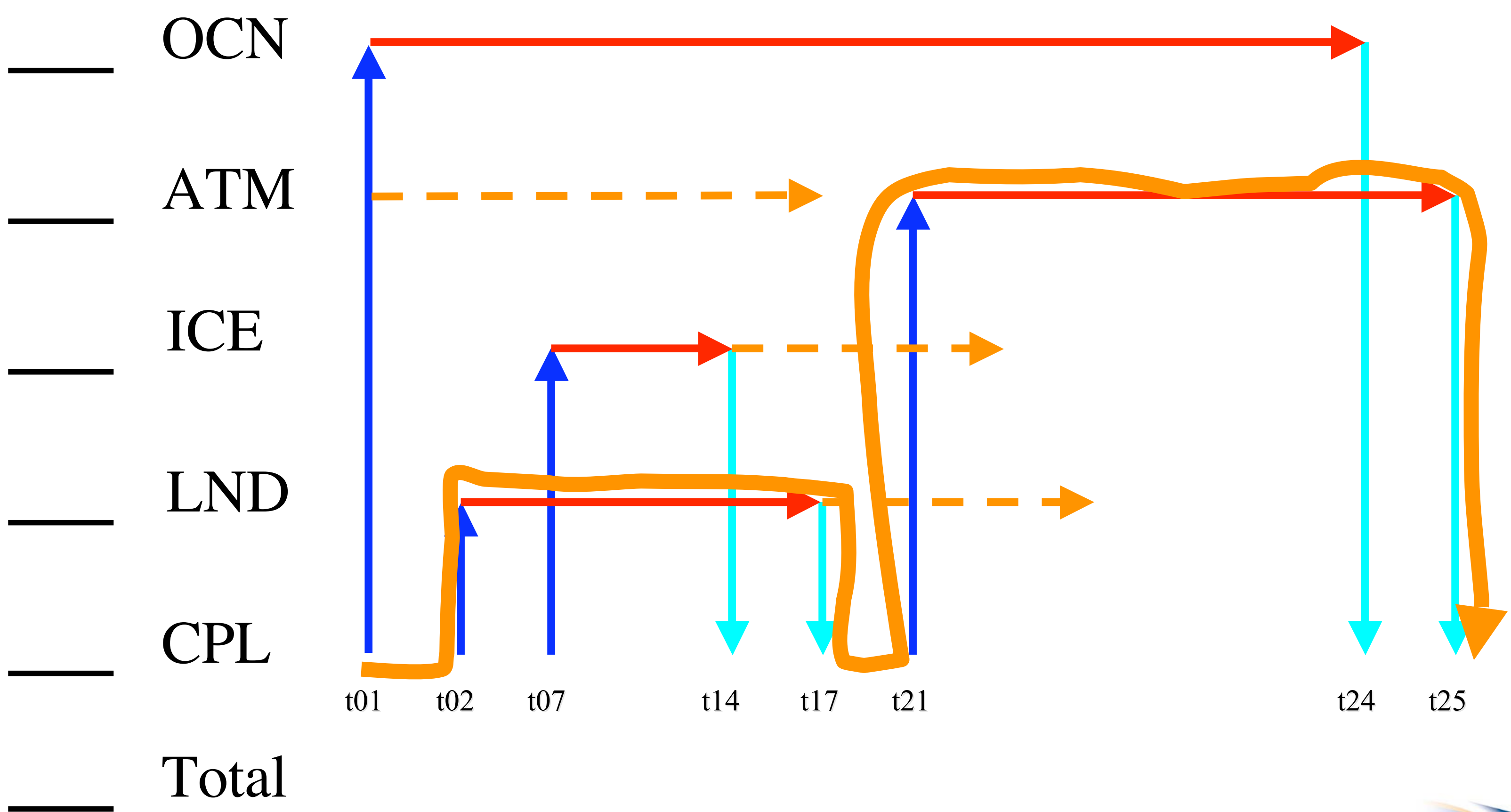
CCSM Version ____________
Machine ____________
Date ____________
Resolution ____________
Config # ____________

Years/day ____________
Years/day/cpu ____________
CPL main time ____________
CPL avg dt ____________



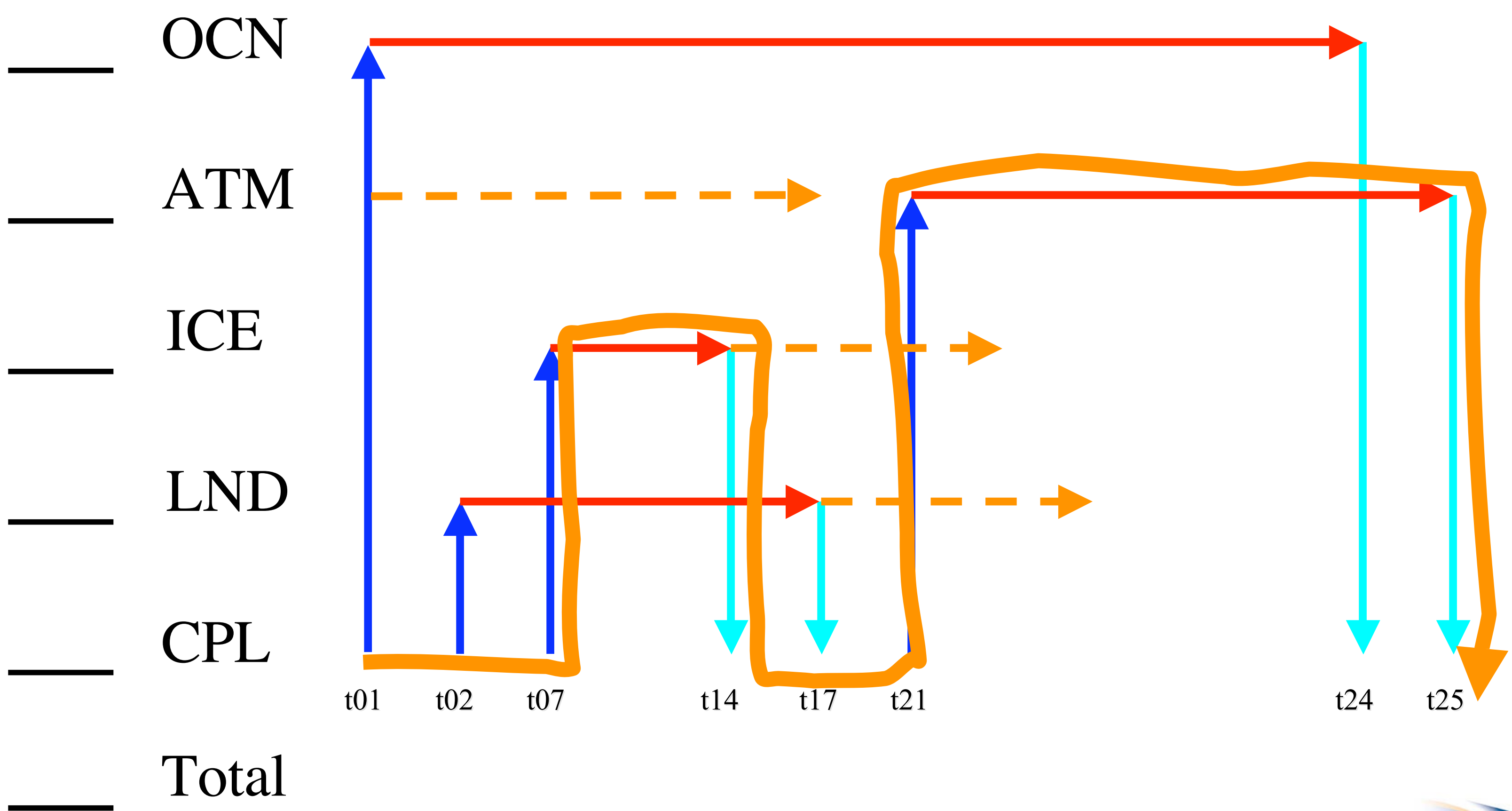
CCSM Version ____________
Machine ____________
Date ____________
Resolution ____________
Config # ____________

Years/day ____________
Years/day/cpu ____________
CPL main time ____________
CPL avg dt ____________



CCSM Version ____________
Machine ____________
Date ____________
Resolution ____________
Config # ____________

Years/day ____________
Years/day/cpu ____________
CPL main time ____________
CPL avg dt ____________



NCAR

Running CCSM: The Basic Steps

- cd ${CCSMROOT}/scripts
- ./create_newcase -case ~/test/T31x3 -mach calgary -res T31_gx3v5 -compset B
- cd ~/test/T31x3
- edit env_run to set run for 10 days. I also usually set INFO_DBUG to 0 and DIAG_OPTION to never (but that's not required).
- edit env_mach.calgary to set DOUT_S to FALSE
- configure -mach calgary
- ${CASE}.calgary.build [this builds and prestages data for the run]
- edit the ${CASE}.calgary.build if you need to set queues, time limits, or accounts for PBS
- qsub ${CASE}.calgary.run
- ${CCSMROOT}/scripts/ccsm_utils/Tools/timing/getTiming.csh -mach calgary -tdir `pwd`

Note: See CCSM User's Guide and CCSM Scripts Tutorial

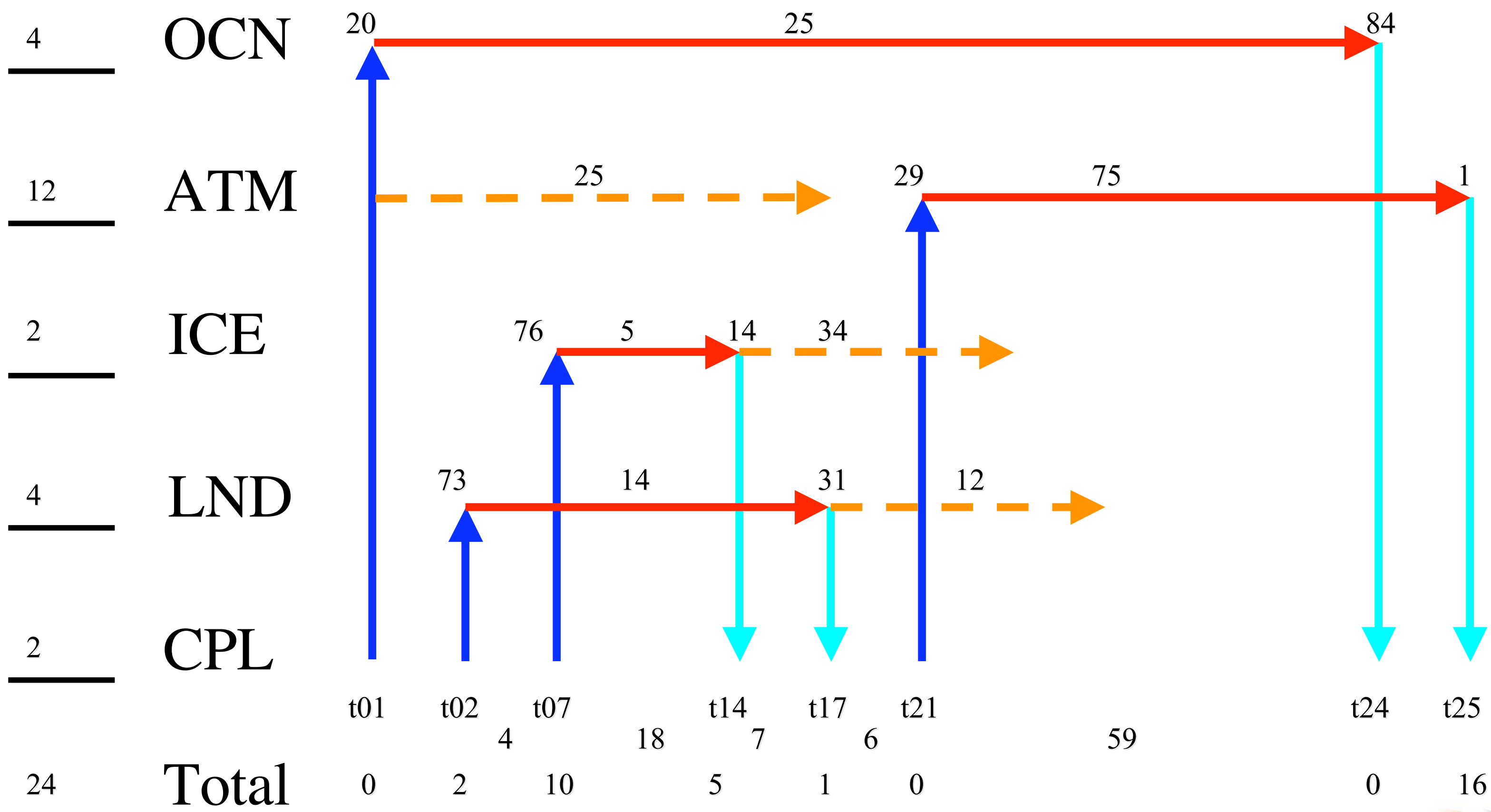
Cray X1

- ORNL's Phoenix
  - Each node has 4 MSPs
  - Queuing in multiples of 4 MSPs
- T31x3 standard run
- Started with 6 nodes (24 MSPs)
- CAM: 12 MPI tasks (12 MSPs)
- Goal: find small configuration
  - Better efficiency
  - Better queue time

NCAR

CCSM Version	M3
Machine	Phoenix
Date	10/16/04
Resolution	T31x3
Config #	12-1

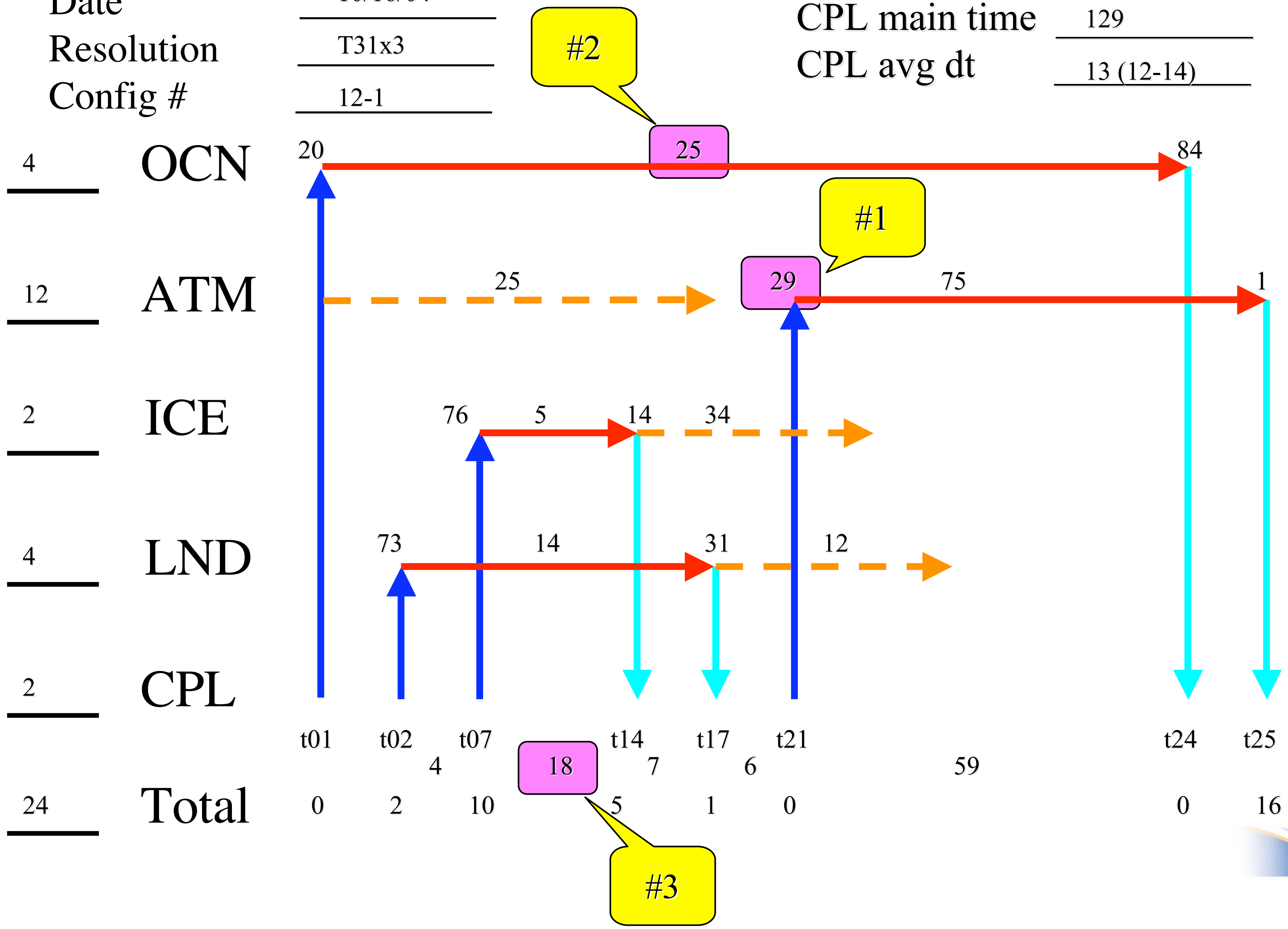
Years/day	18.4
Years/day/cpu	0.7667
CPL main time	129
CPL avg dt	13 (12-14)



NCAR

CCSM Version M3
Machine Phoenix
Date 10/16/04
Resolution T31x3
Config # 12-1

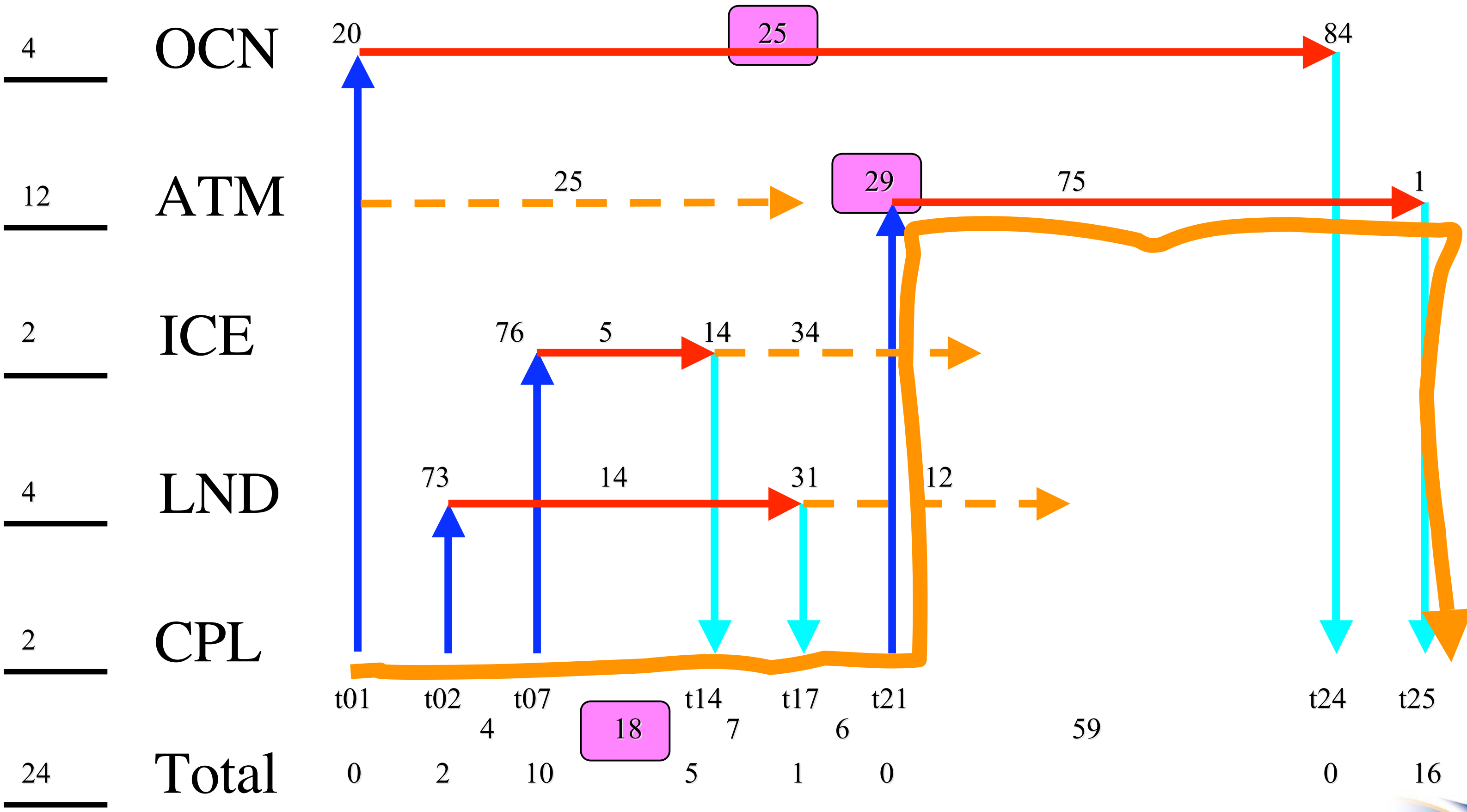
Years/day 18.4
Years/day/cpu 0.7667
CPL main time 129
CPL avg dt 13 (12-14)



NCAR

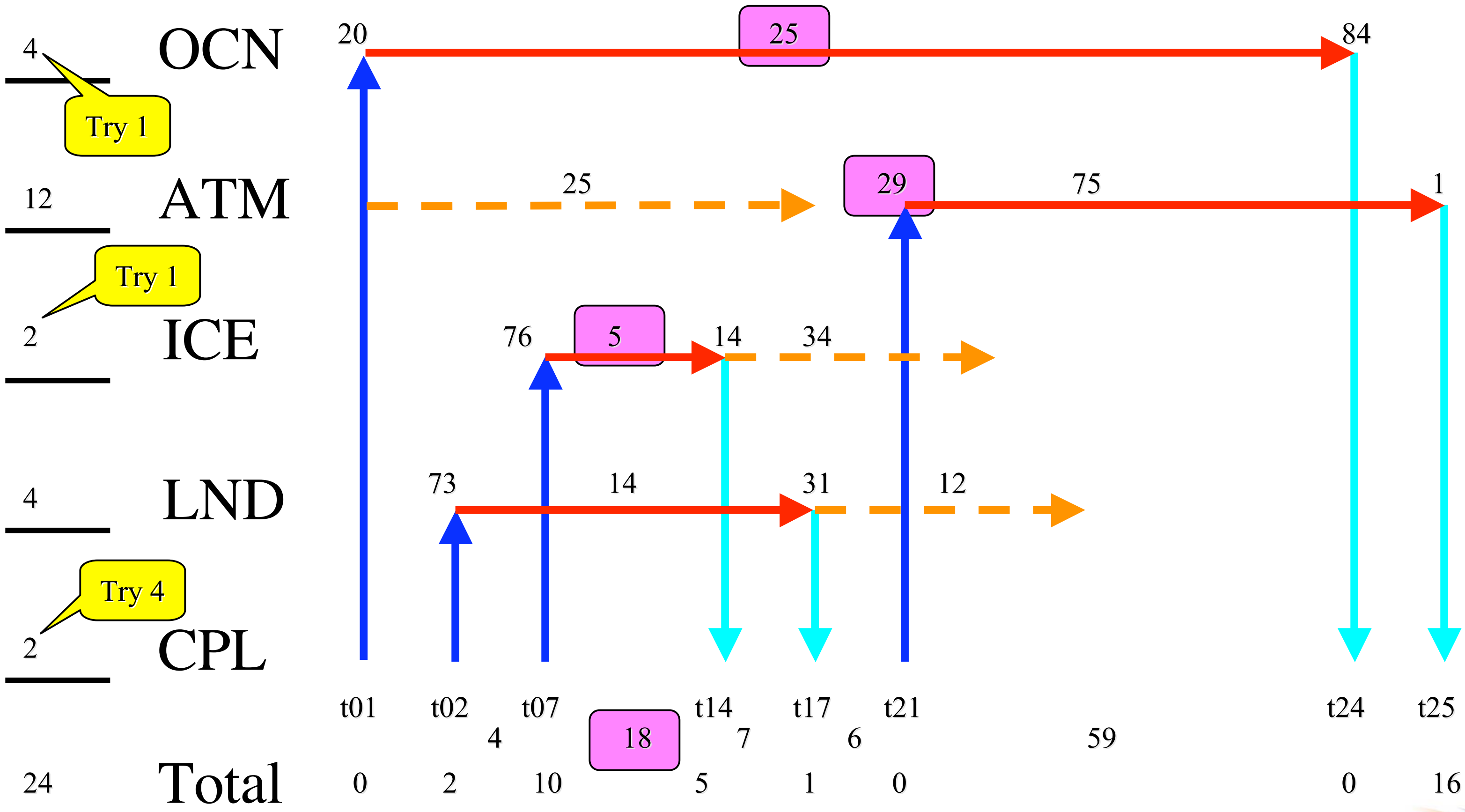
CCSM Version	M3
Machine	Phoenix
Date	10/16/04
Resolution	T31x3
Config #	12-1

Years/day	18.4
Years/day/cpu	0.7667
CPL main time	129
CPL avg dt	13 (12-14)



CCSM Version	M3
Machine	Phoenix
Date	10/16/04
Resolution	T31x3
Config #	12-1

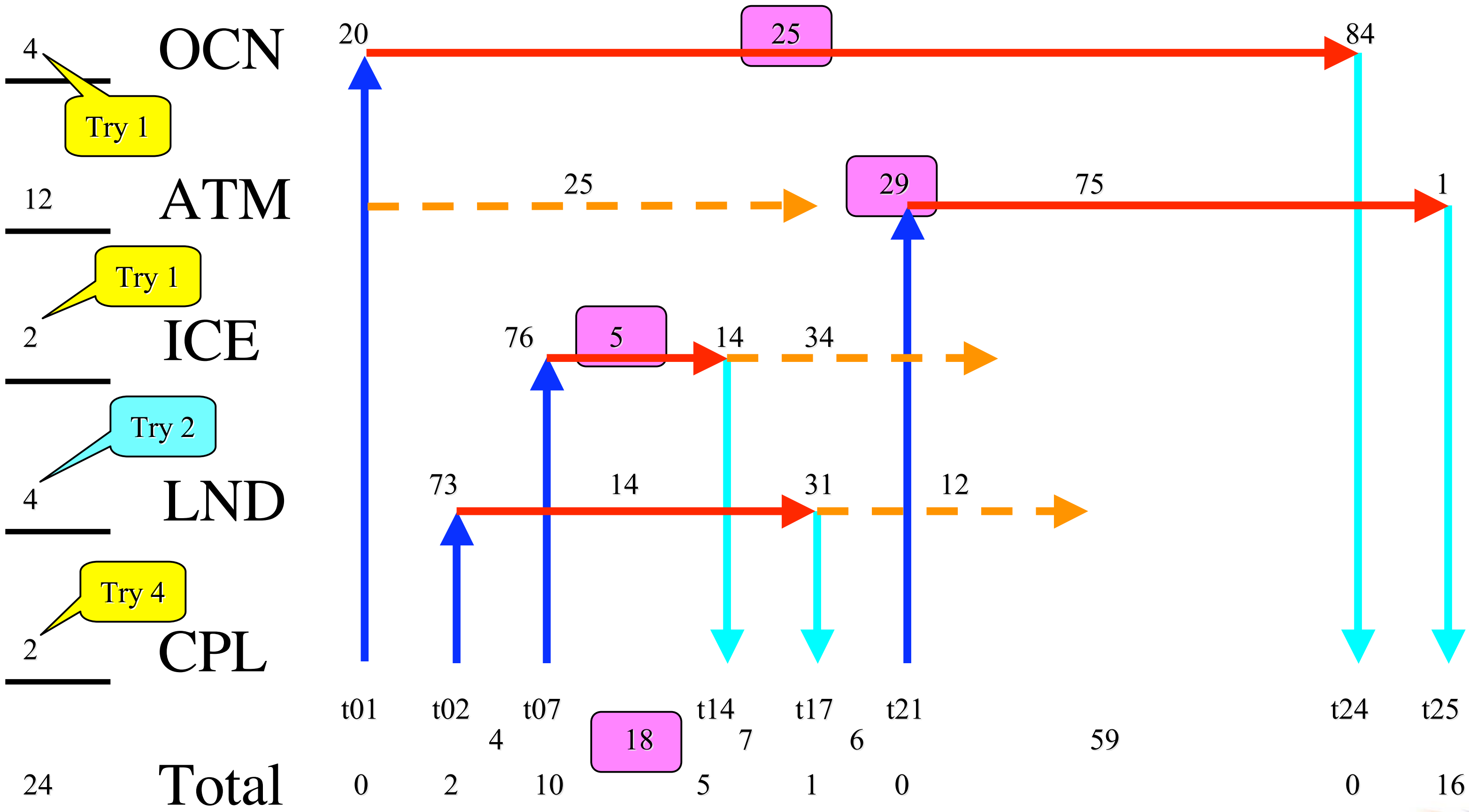
Years/day	18.4
Years/day/cpu	0.7667
CPL main time	129
CPL avg dt	13 (12-14)



NCAR

CCSM Version	M3
Machine	Phoenix
Date	12/9/04
Resolution	T31x3
Config #	12-1

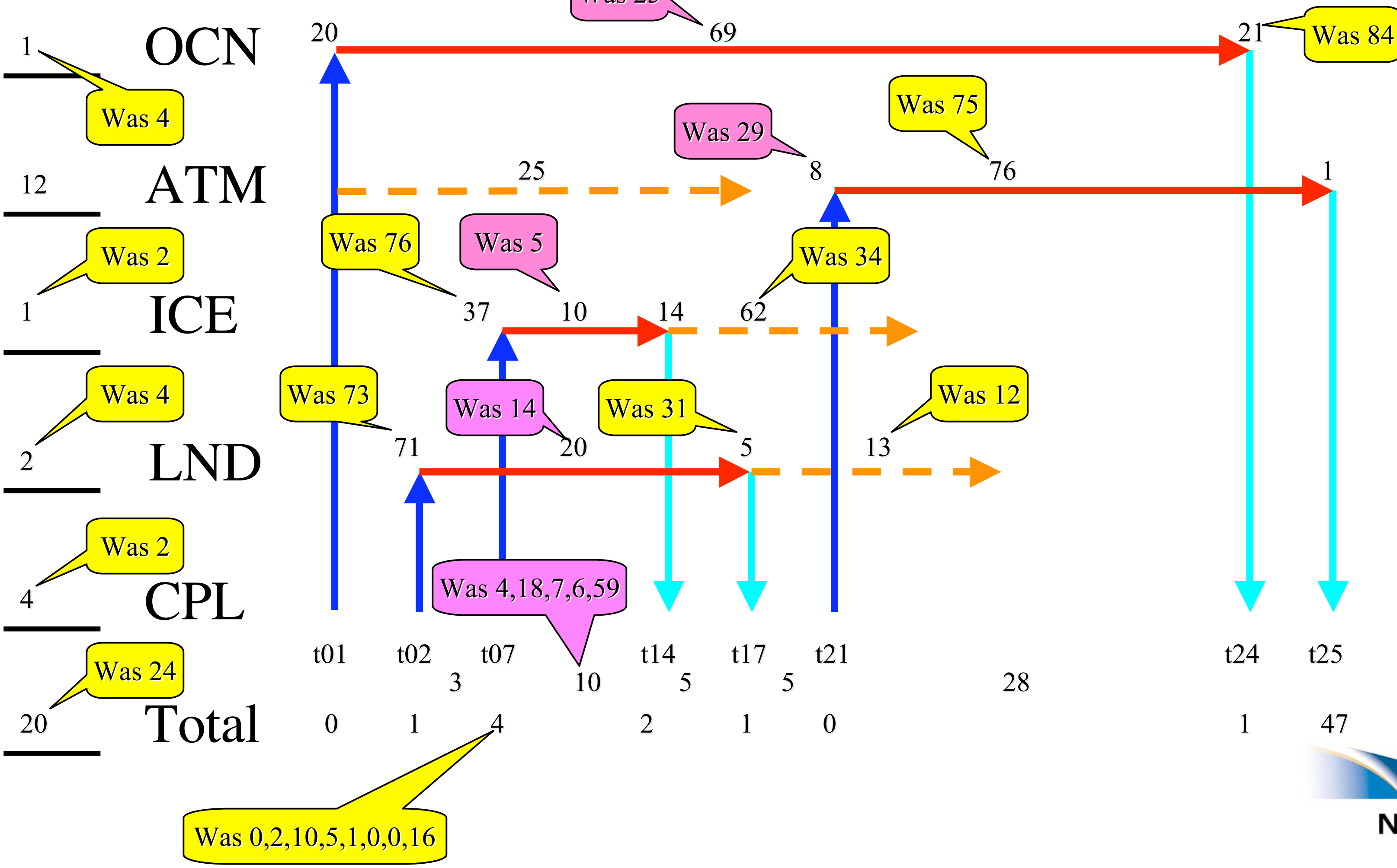
Years/day	18.4
Years/day/cpu	0.7667
CPL main time	129
CPL avg dt	13 (12-14)



NCAR

CCSM Version	M3
Machine	Phoenix
Date	12/9/04
Resolution	T31x3
Config #	12-2

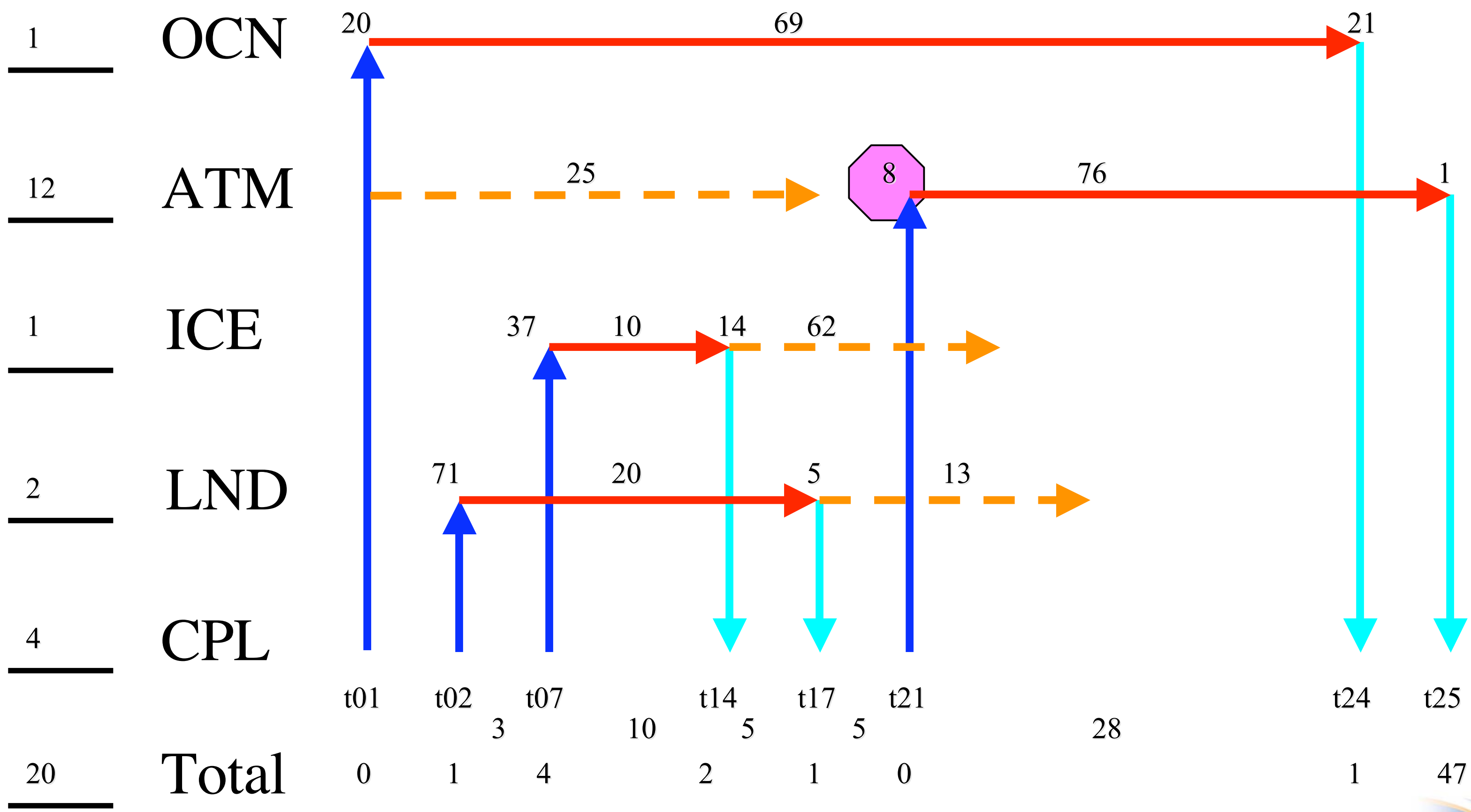
Years/day	21.81	Was 18.4
Years/day/cpu	1.0905	0.7667
CPL main time	108	129
CPL avg dt	11	13



NCAR

CCSM Version	M3
Machine	Phoenix
Date	12/9/04
Resolution	T31x3
Config #	12-2

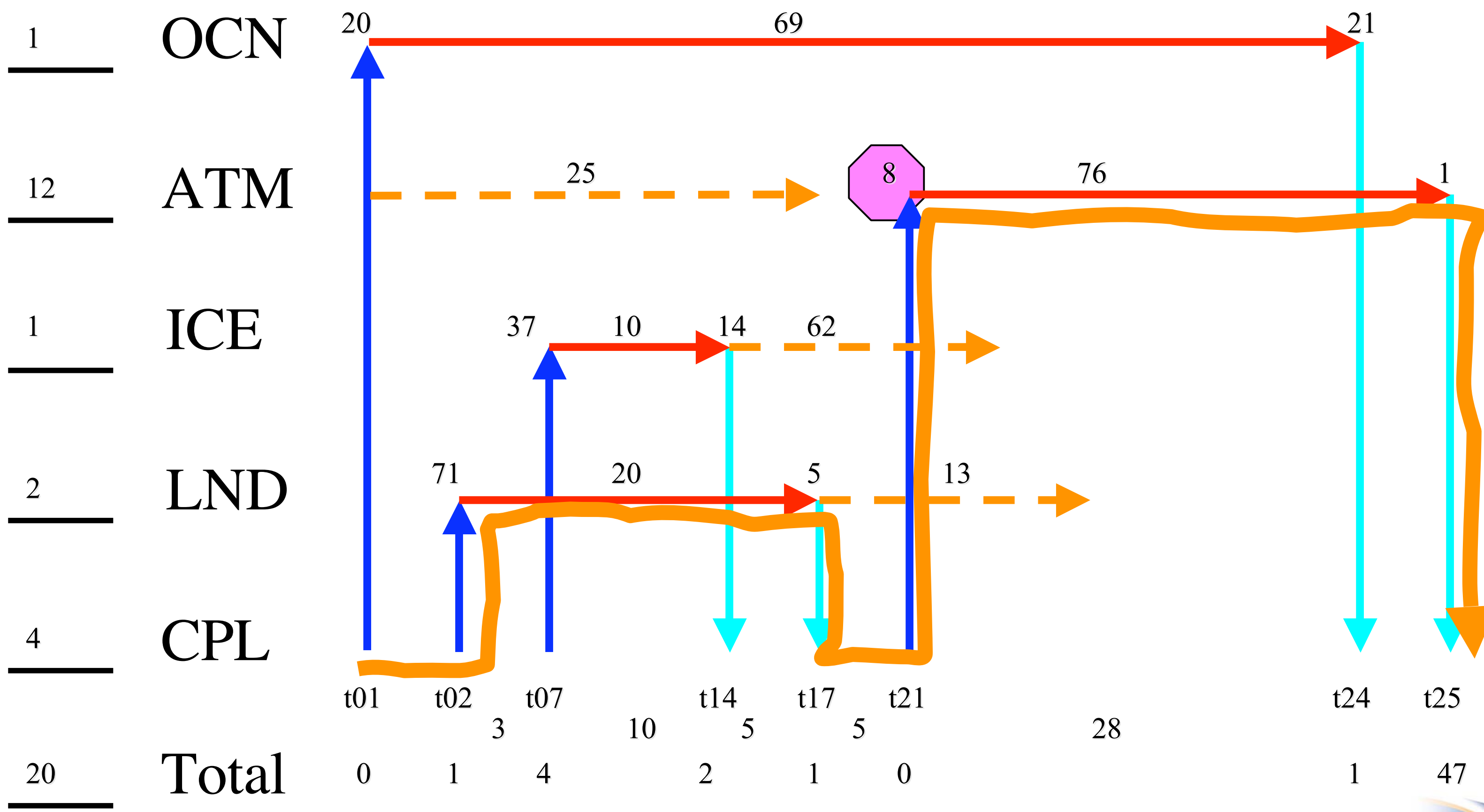
Years/day	21.81
Years/day/cpu	1.0905
CPL main time	108
CPL avg dt	11



NCAR

CCSM Version	M3
Machine	Phoenix
Date	12/9/04
Resolution	T31x3
Config #	12-2

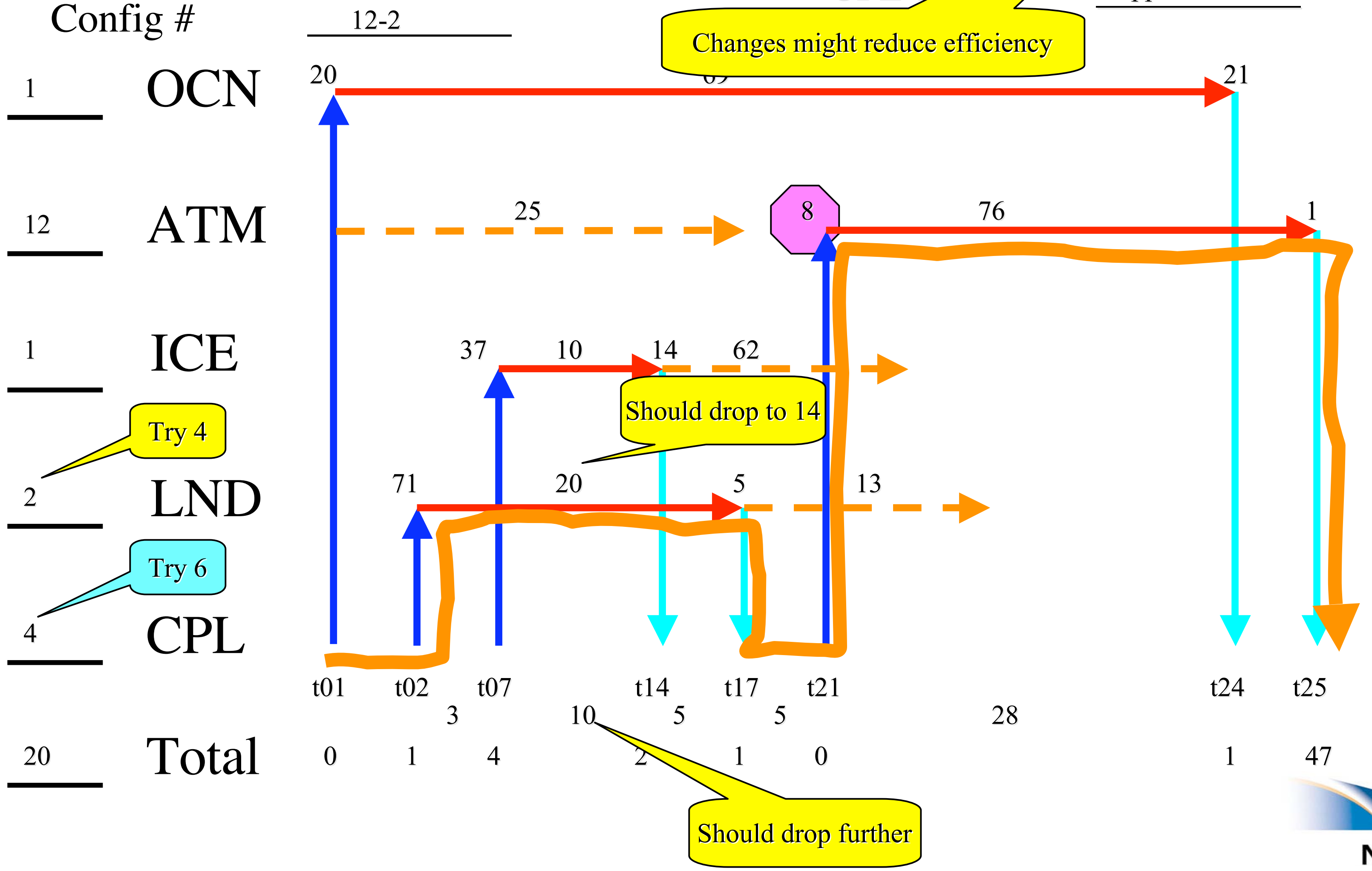
Years/day	21.81
Years/day/cpu	1.0905
CPL main time	108
CPL avg dt	11



NCAR

CCSM Version	M3
Machine	Phoenix
Date	12/9/04
Resolution	T31x3
Config #	12-2

Years/day	21.81
Years/day/cpu	1.0905
CPL main tim	108
CPL av	11



NCAR

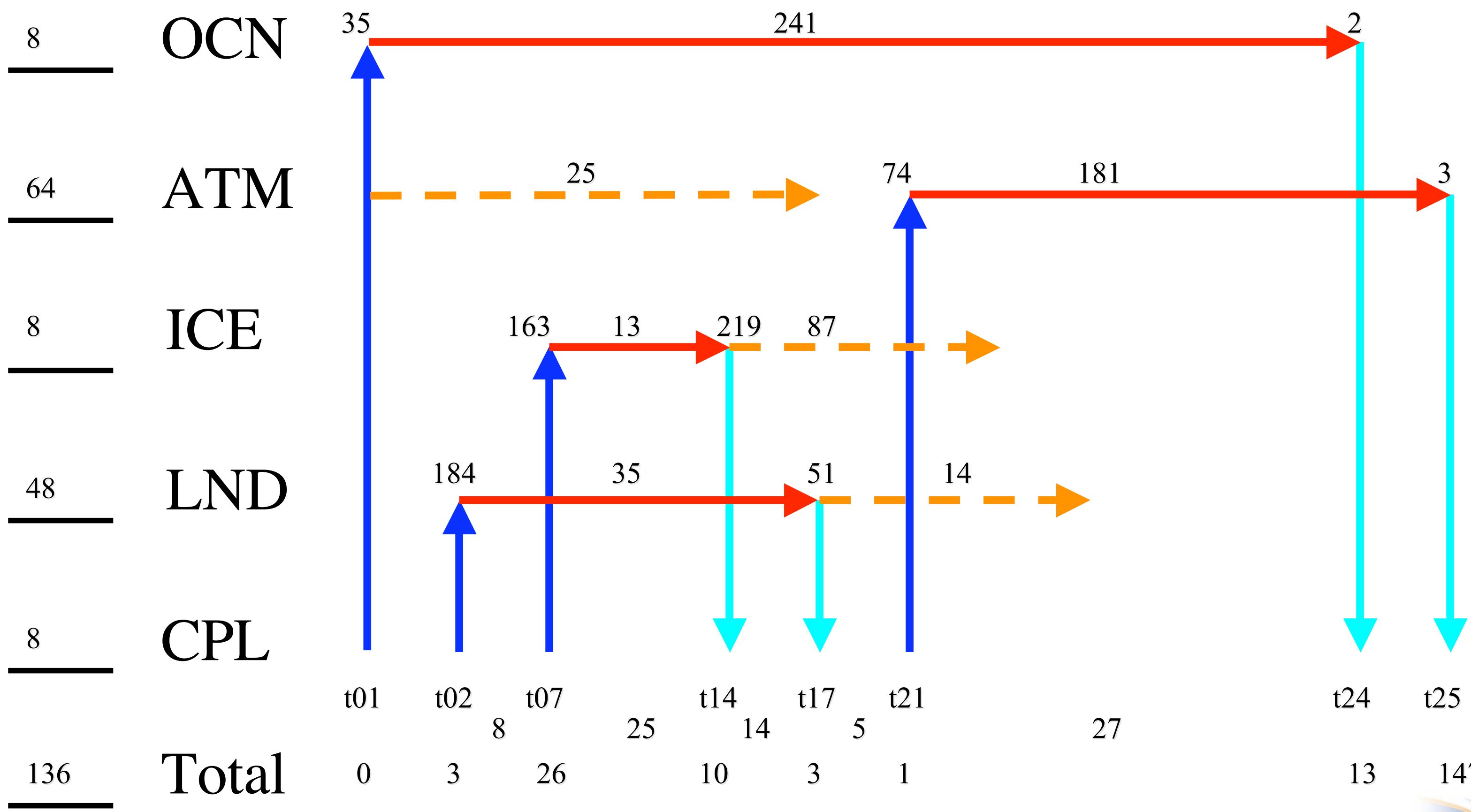
Cray X1

- ORNL's Phoenix
  - Each node has 4 MSPs
  - Queuing in multiples of 4 MSPs
- T85x1 standard run
- Started with 34 nodes (136 MSPs)
- CAM: 64 MPI tasks (64 MSPs)
- Goal: Looking for compromise of years per day and efficiency

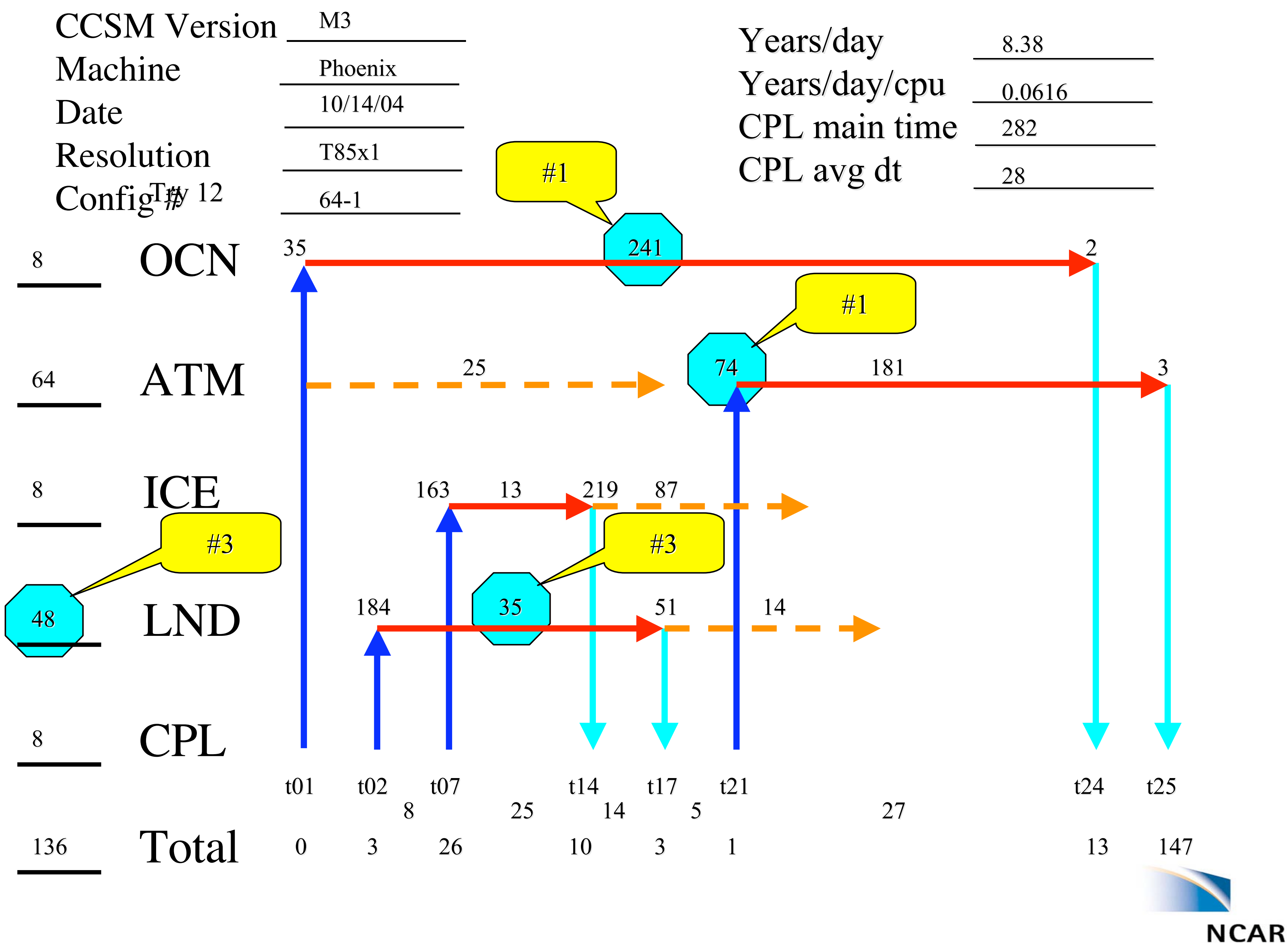
NCAR

CCSM Version M3
Machine Phoenix
Date 10/14/04
Resolution T85x1
Config # 64-1

Years/day 8.38
Years/day/cpu 0.0616
CPL main time 282
CPL avg dt 28

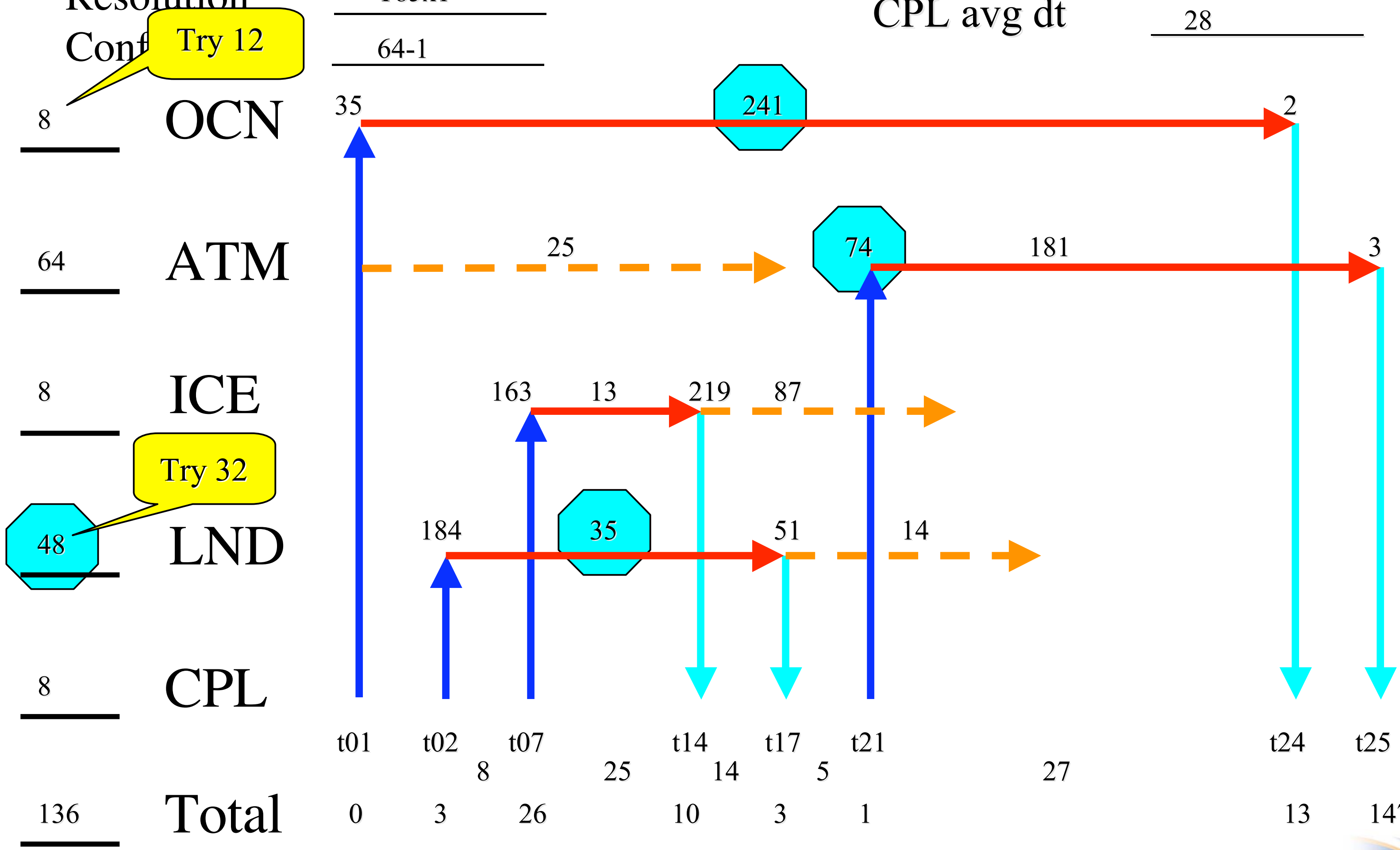


NCAR

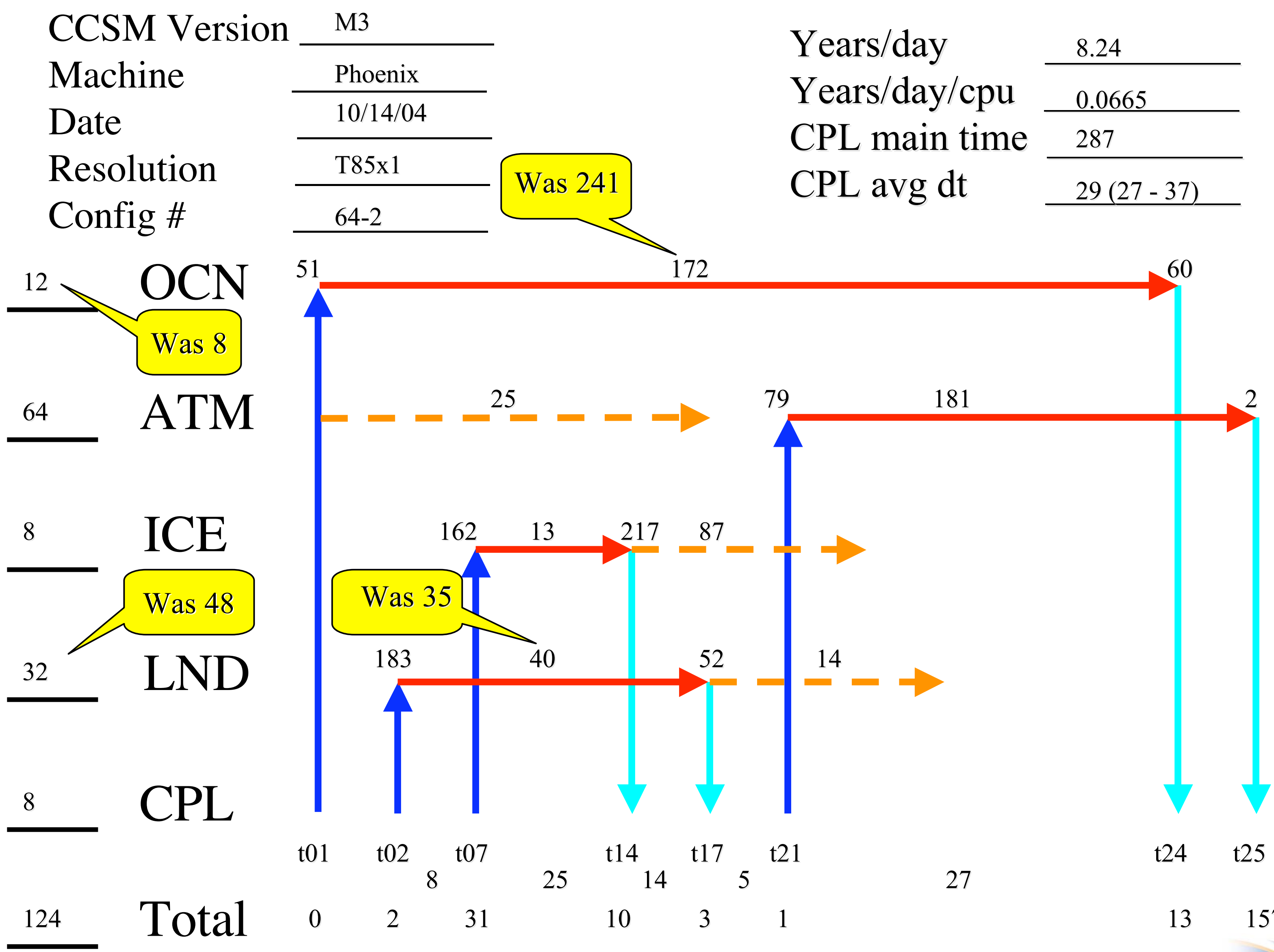

CCSM Version M3
Machine Phoenix
Date 10/14/04
Resolution T85x1
Config #Try 12 64-1
Years/day 8.38
Years/day/cpu 0.0616
CPL main time 282
CPL avg dt 28
8 OCN
64 ATM
8 ICE
48 LND
8 CPL
136 Total
35
241
#1
2
25
74
#1
181
3
163
13
219
87
#3
184
35
#3
51
14
t01
t02
t07
t14
t17
t21
t24
t25
8
25
14
5
27
0
3
26
10
3
1
13
147
NCAR

CCSM Version	M3
Machine	Phoenix
Date	10/14/04
Resolution	T85x1
Conf	64-1

Years/day	8.38
Years/day/cpu	0.0616
CPL main time	282
CPL avg dt	28



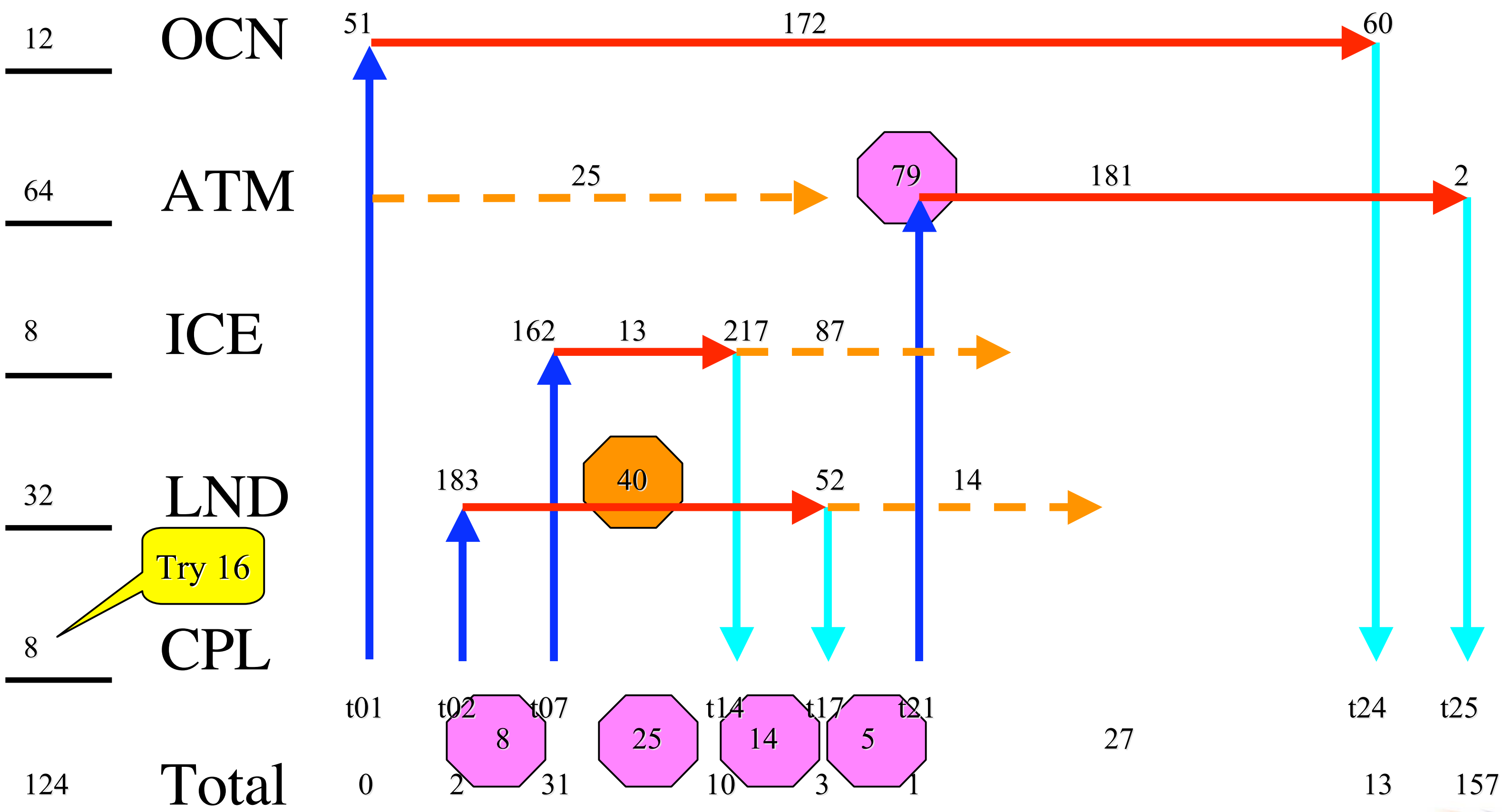
NCAR



CCSM Version M3
Machine Phoenix
Date 10/14/04
Resolution T85x1
Config # 64-2
Years/day 8.24
Years/day/cpu 0.0665
CPL main time 287
CPL avg dt 29 (27 - 37)
Was 241
Was 8
Was 48
Was 35
12 OCN
64 ATM
8 ICE
32 LND
8 CPL
124 Total
51
172
60
25
79
181
2
162
13
217
87
183
40
52
14
t01
t02
t07
t14
t17
t21
t24
t25
8
25
14
5
27
0
2
31
10
3
1
13
157

CCSM Version M3
Machine Phoenix
Date 10/14/04
Resolution T85x1
Config # 64-2

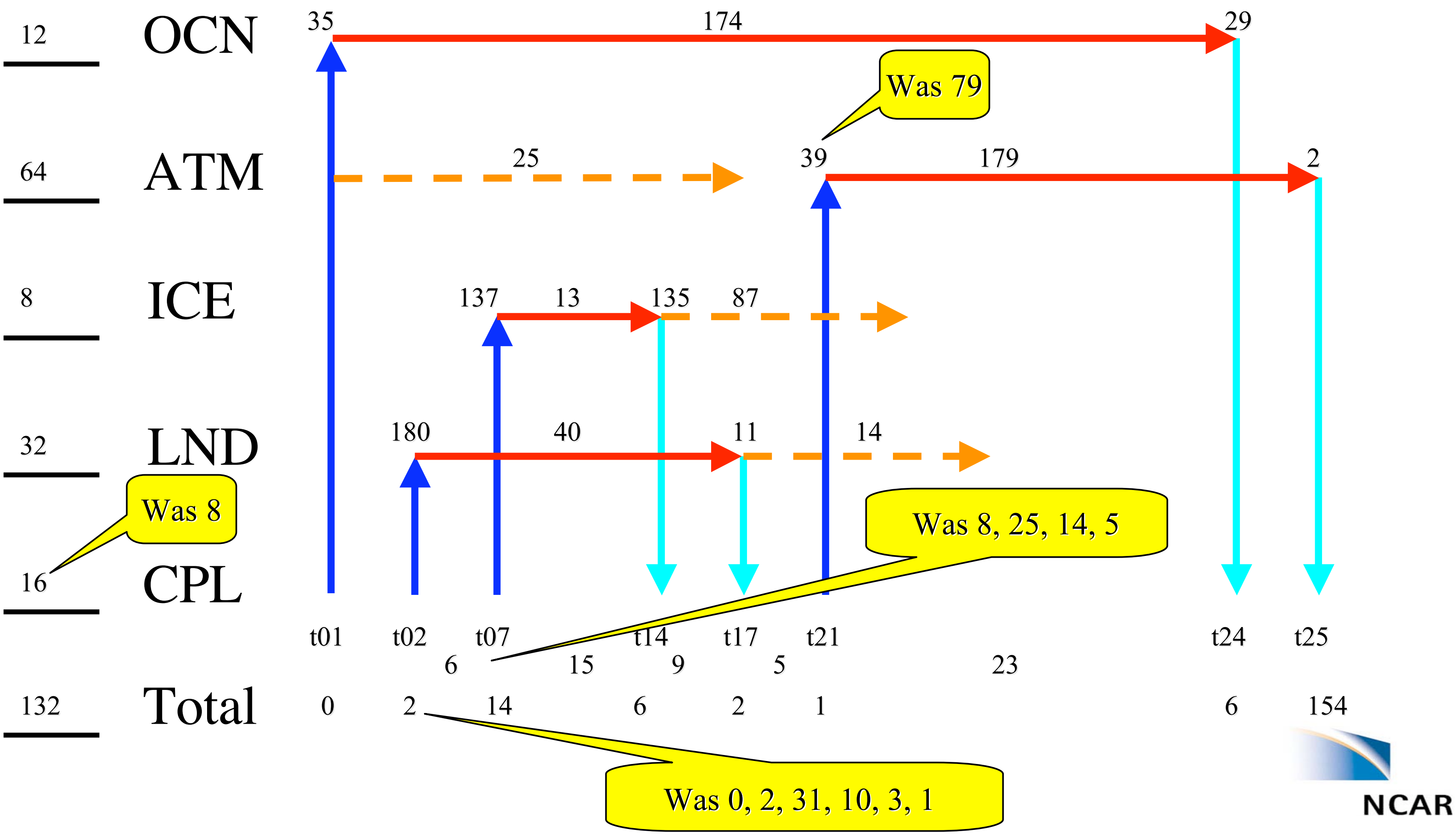
Years/day 8.24
Years/day/cpu 0.0665
CPL main time 287
CPL avg dt 29 (27 - 37)



NCAR

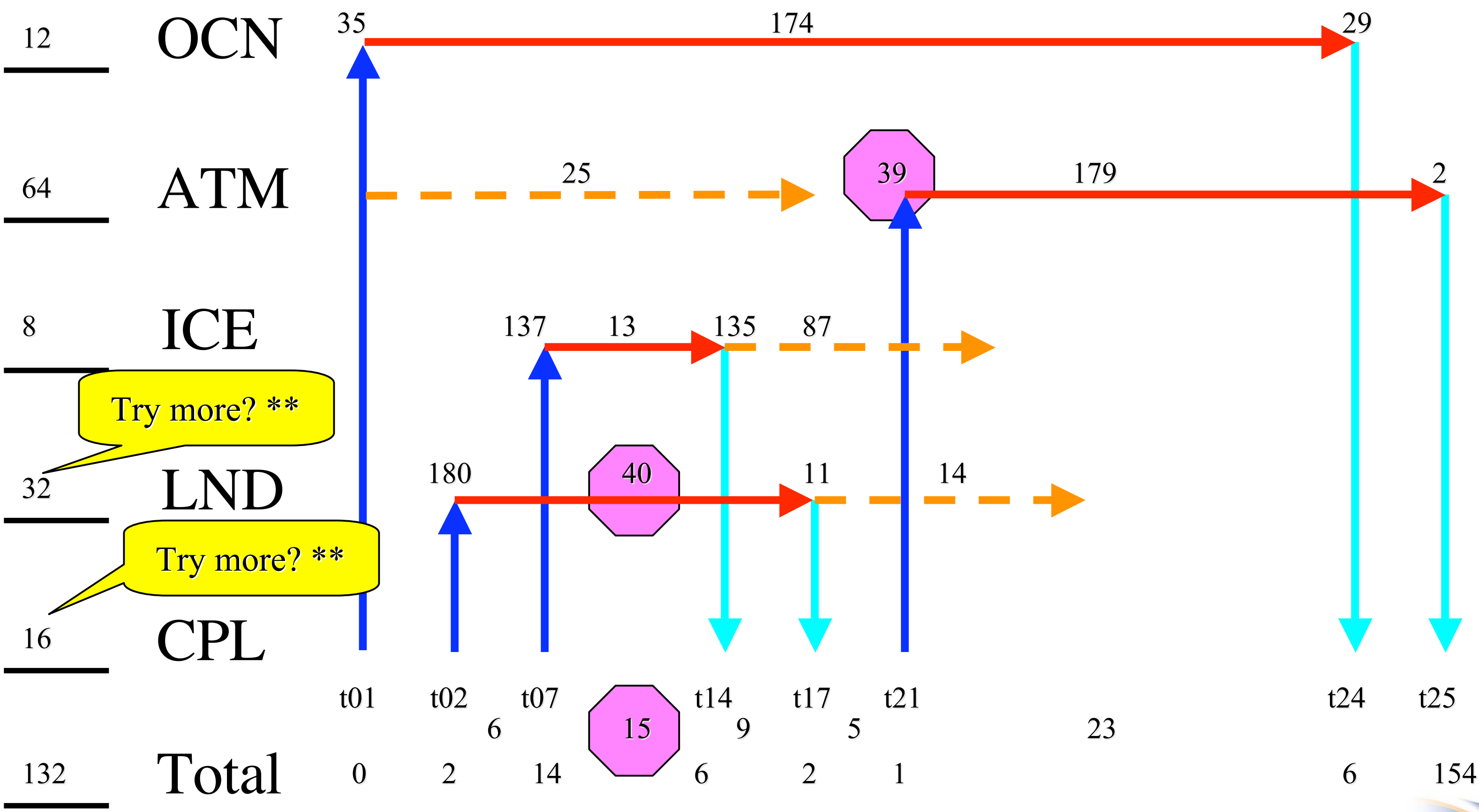
CCSM Version	M3
Machine	Phoenix
Date	10/14/04
Resolution	T85x1
Config #	64-3

Years/day	9.73	Was 8.24
Years/day/cpu	0.0737	0.0665
CPL main time	243	287
CPL avg dt	24 (22 - 31)	29



CCSM Version	M3
Machine	Phoenix
Date	10/14/04
Resolution	T85x1
Config #	64-3

Years/day	9.73
Years/day/cpu	0.0737
CPL main time	243
CPL avg dt	24 (22 - 31)



** From Previous Tests we know that 48 LNDs only reduces from 40 to 35
Need to speed up LND and CPL components

NCAR

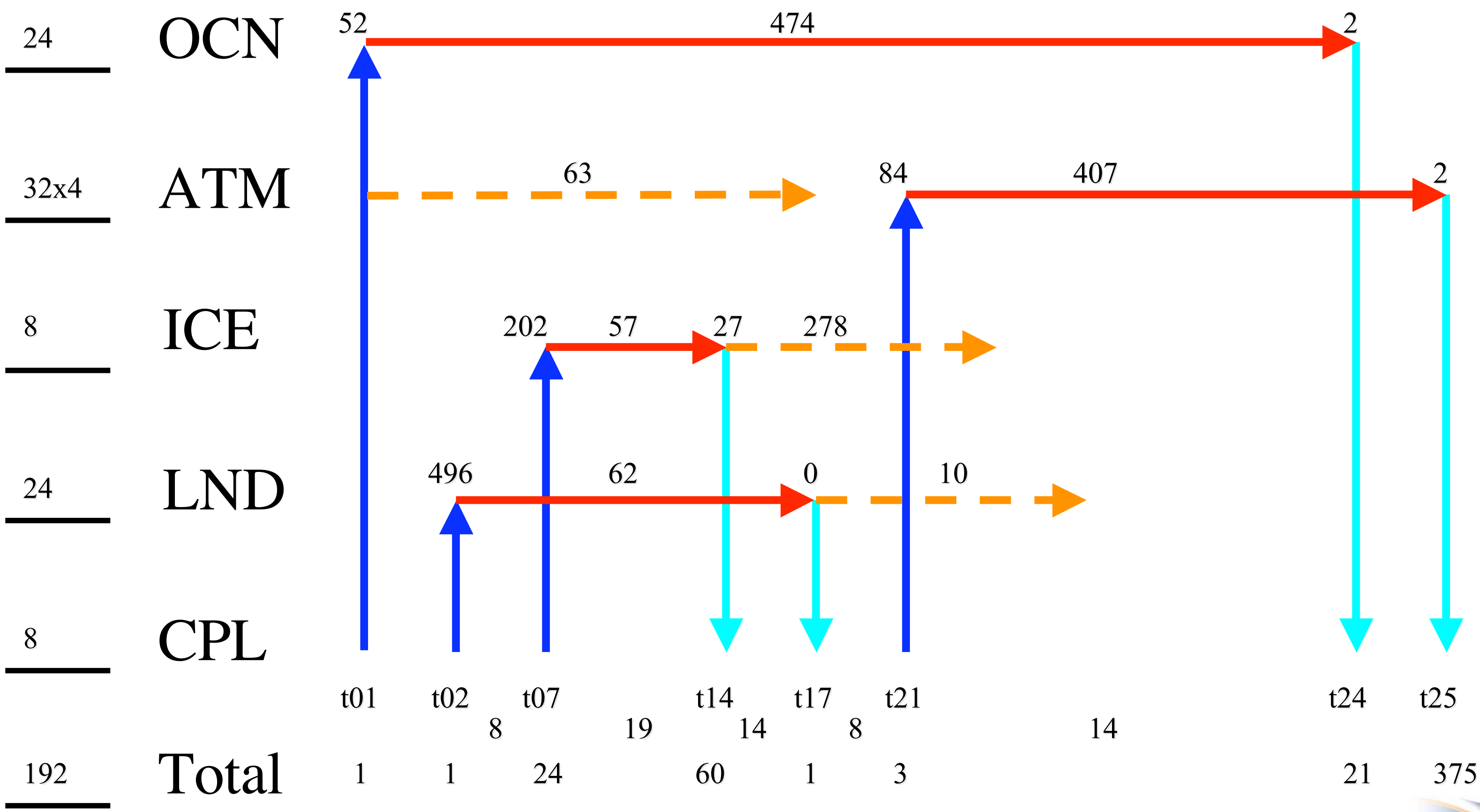
IBM 8 and 32 Way Example

- NCAR'S bluesky
  - Each node has 8 processors
    - More network connections
  - Each node has 32 processors
    - Fast messaging for 32 processors on node
  - Colony switch
  - Job queuing in whole node multiples
- T85x1 standard run
- 24 8way nodes or 6 32way nodes (192 CPUs) (common IPCC job size)
- CAM: 32 MPI tasks, 4 threads per MPI (128 CPUs)

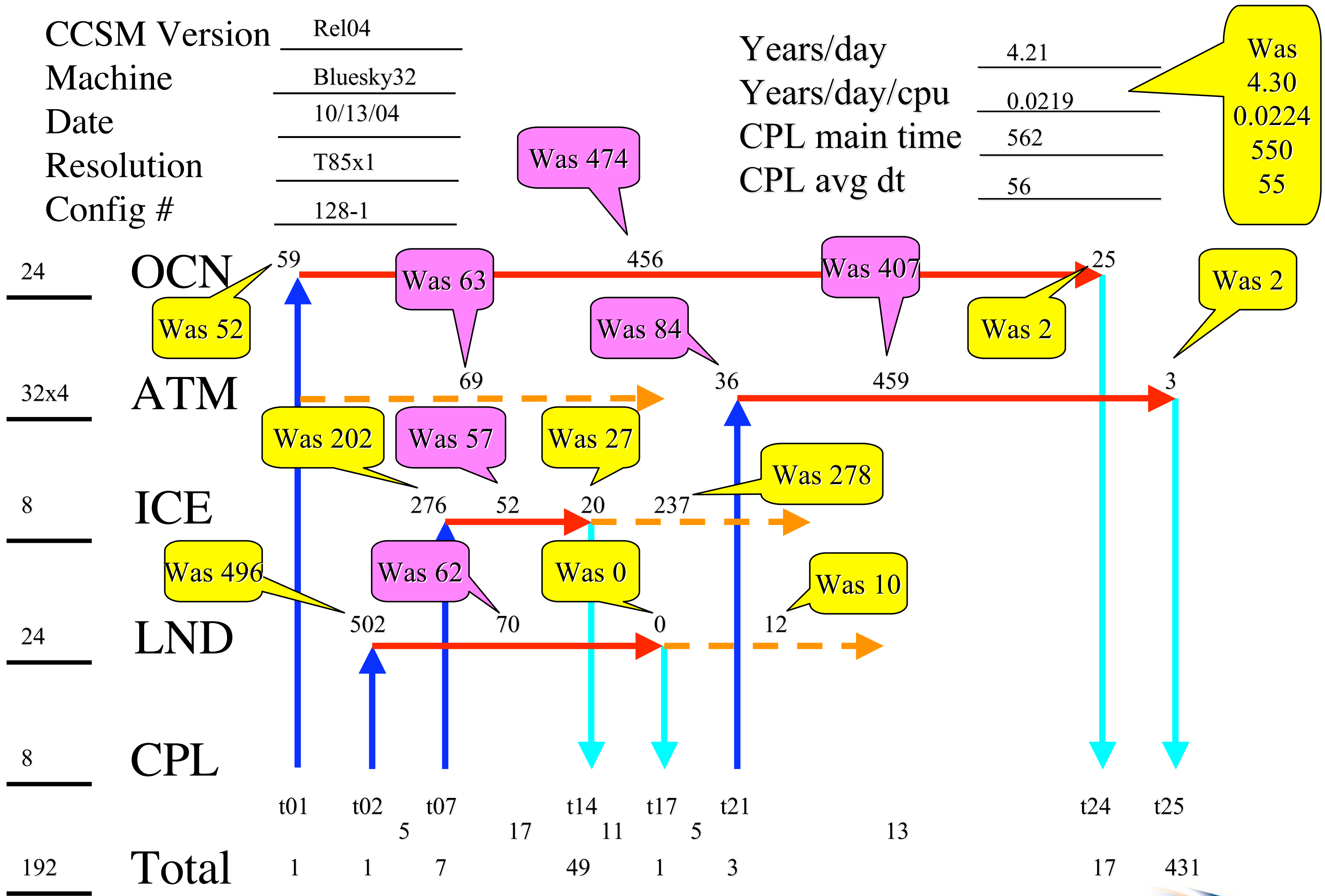
NCAR

CCSM Version	Rel04
Machine	Bluesky8
Date	10/13/04
Resolution	T85x1
Config #	128-1

Years/day	4.30
Years/day/cpu	0.0224
CPL main time	550
CPL avg dt	55



NCAR



CCSM Version Rel04
Machine Bluesky32
Date 10/13/04
Resolution T85x1
Config # 128-1
Years/day 4.21
Years/day/cpu 0.0219
CPL main time 562
CPL avg dt 56
Was 4.30 0.0224 550 55
Was 474
24 OCN
59
Was 63
456
Was 407
25
Was 52
Was 84
Was 2
Was 2
32x4 ATM
69
36
459
3
Was 202
Was 57
Was 27
Was 278
8 ICE
276
52
20
237
Was 496
Was 62
Was 0
Was 10
24 LND
502
70
0
12
8 CPL
t01
t02
t07
t14
t17
t21
t24
t25
5
17
11
5
13
192 Total
1
1
7
49
1
3
17
431

NCAR

8 way vs 32 way - What Happened?

- Allocation of processes to processors
  - set COMPONENTS = ($COMP_CPL $COMP_ICE $COMP_LND $COMP_OCN $COMP_ATM)

    32way: (8c8i16l),(8l,24o),4x(32a)

    8way: (8c),(8i),3x(8l),3x(8o),16x(8a)
  - Anything wrong? Anything better?
    - Land split across two nodes
  - Might make better use of 32 way

    32way: (8c24l),(8i,24o),4x(32a)

    32way: (8c24o),(8i,24l),4x(32a)

    Which? Why?

NCAR

CCSM Hybrid Example on IBM

- Thunder is IBM system with four 16 way nodes and Federation switches
- We typically run with 4 CAM threads on IBM systems
- Tried 0, 4, and 8 threads keeping total number of CAM processors constant 48

Num CAM MPI tasks	Num CAM Threads	Coupled Years per Day
48	0	18.87
12	4	20.34
6	8	21.94

NCAR

For Further Information

- CCSM web pages
  - http://www.ccsm.ucar.edu/ccsm3
  - http://www.ccsm.ucar.edu/support_model
    - See CCSM User's Guide
    - See Scripts Tutorial
  - http://www.ccsm.ucar.edu/support_model/mach_support.html
- CCSM Bulletin Board
  - http://bb.cgd.ucar.edu
- gcarr@ucar.edu

NCAR