# Massively Parallel Computing for NWP and Climate

**George Mozdzynski**

**George.Mozdzynski@ecmwf.int**

ECMWF

# Outline

- Parallel computing?
- Massively parallel?
- Types of computer
- Parallel Computers today
- Challenges in parallel computing
- Parallel Programming Languages
- OpenMP, OpenACC and MPI

# What is Parallel Computing?

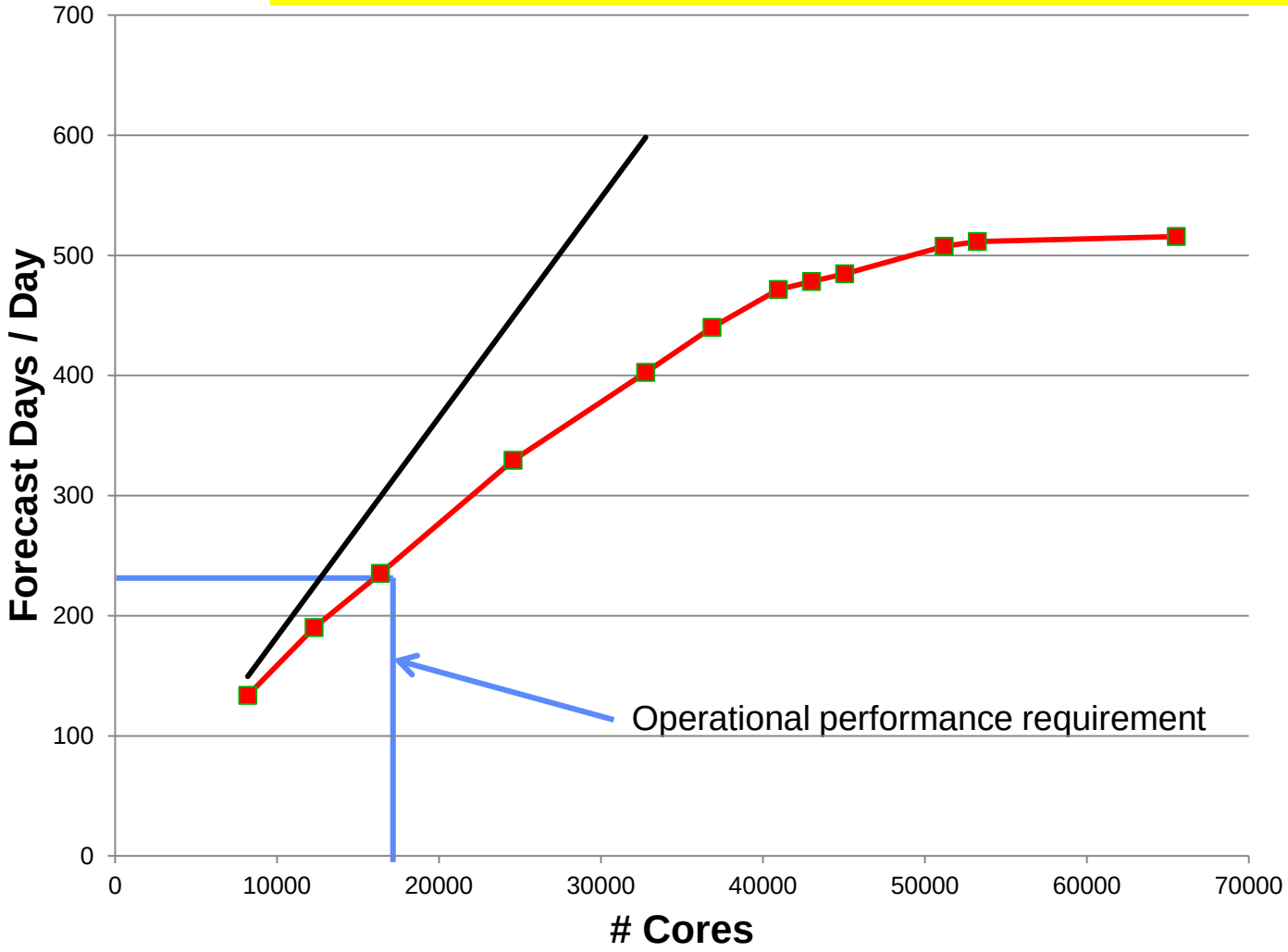
*The simultaneous use of more than one processor or computer to solve a problem*

# Why do we need Parallel Computing?

- **Serial computing is too slow**
- **Need for large amounts of memory not accessible by a single processor**

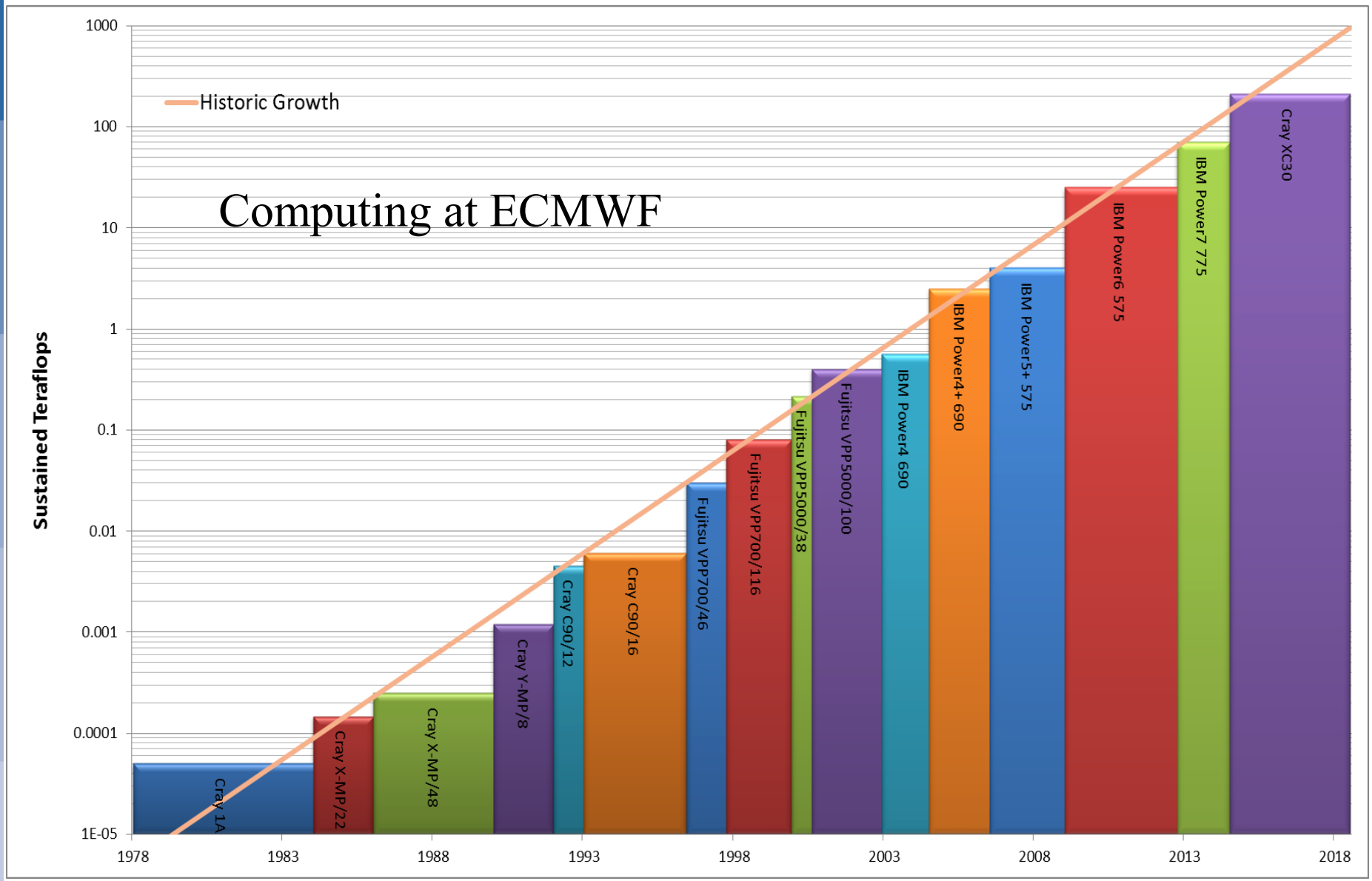
# T2047 IFS global model (10 km) performance on CRAY XE6, 2012

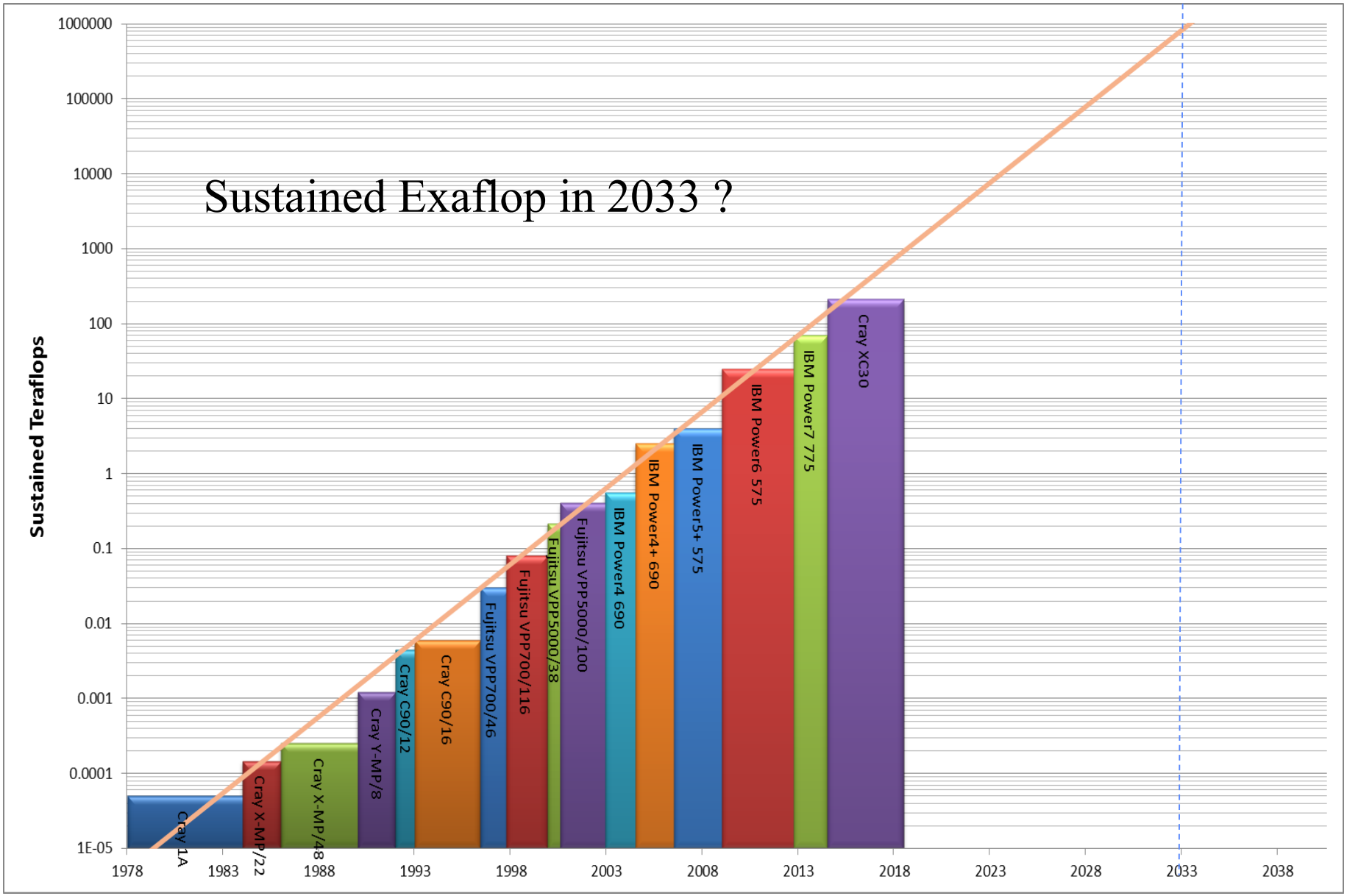
10 day forecast in 1 hour = 240 forecast days / day



# Measuring Performance

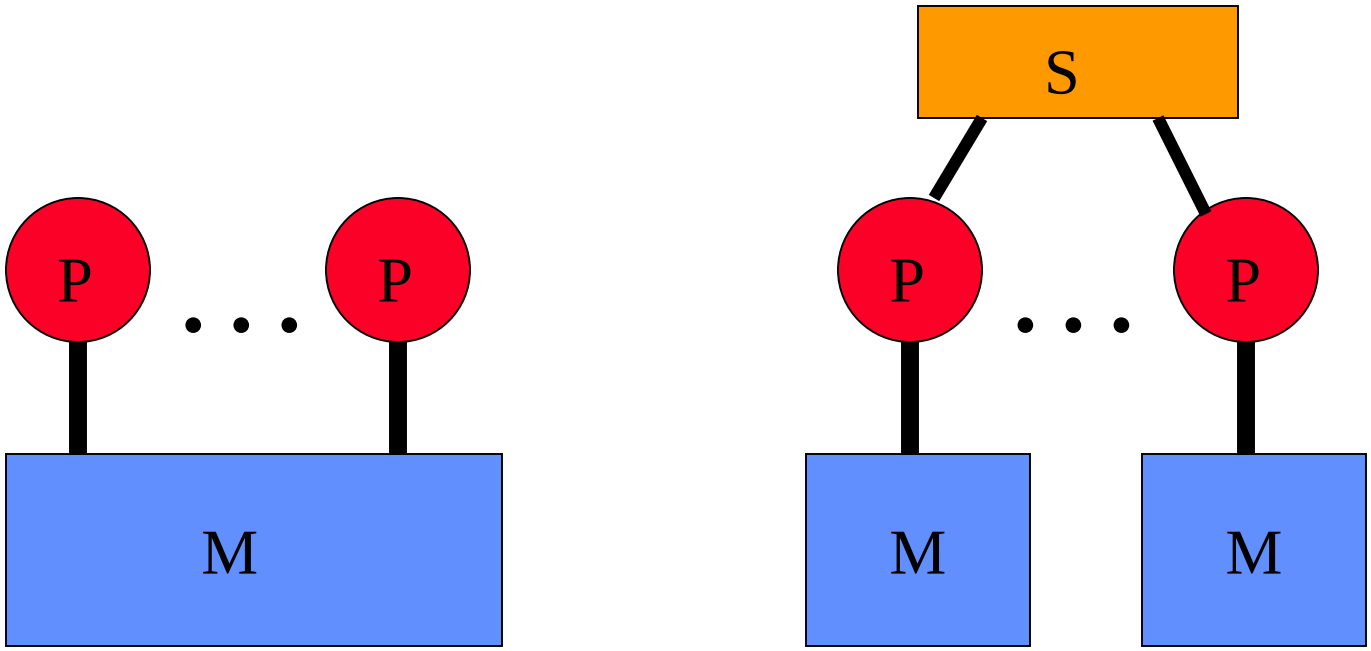
- **Wall Clock**
- **Floating point operations per second (FLOPS or FLOP/S)**
  - **Peak (Hardware), Sustained (Application)**
- **SI prefixes**
  - **Mega Mflops 10**6**
  - **Giga Gflops 10**9**
  - **Tera Tflops 10**12**
  - **Peta Pflops 10**15 ECMWF: 2 * 1.79 Pflops peak (XC-30)**
  - **Exa, Zetta, Yotta**
- **Instructions per second, Mips, etc,**
- **Transactions per second (Databases)**


Computing at ECMWF
Historic Growth
Sustained Teraflops
1000
100
10
1
0.1
0.01
0.001
0.0001
1E-05
1978
1983
1988
1993
1998
2003
2008
2013
2018
Cray 1A
Cray X-MP/22
Cray X-MP/48
Cray Y-MP/8
Cray C90/12
Cray C90/16
Fujitsu VPP700/46
Fujitsu VPP700/116
Fujitsu VPP5000/38
Fujitsu VPP5000/100
IBM Power4 690
IBM Power4+ 690
IBM Power5+ 575
IBM Power6 575
IBM Power7 775
Cray XC30


Sustained Exaflop in 2033 ?
Sustained Teraflops
1000000
100000
10000
1000
100
10
1
0.1
0.01
0.001
0.0001
1E-05
1978
1983
1988
1993
1998
2003
2008
2013
2018
2023
2028
2033
2038
Cray 1A
Cray X-MP/22
Cray X-MP/48
Cray Y-MP/8
Cray C90/12
Cray C90/16
Fujitsu VPP700/46
Fujitsu VPP700/116
Fujitsu VPP5000/38
Fujitsu VPP5000/100
IBM Power4 690
IBM Power4+ 690
IBM Power5+ 575
IBM Power6 575
IBM Power7 775
Cray XC30

# Types of Parallel Computer

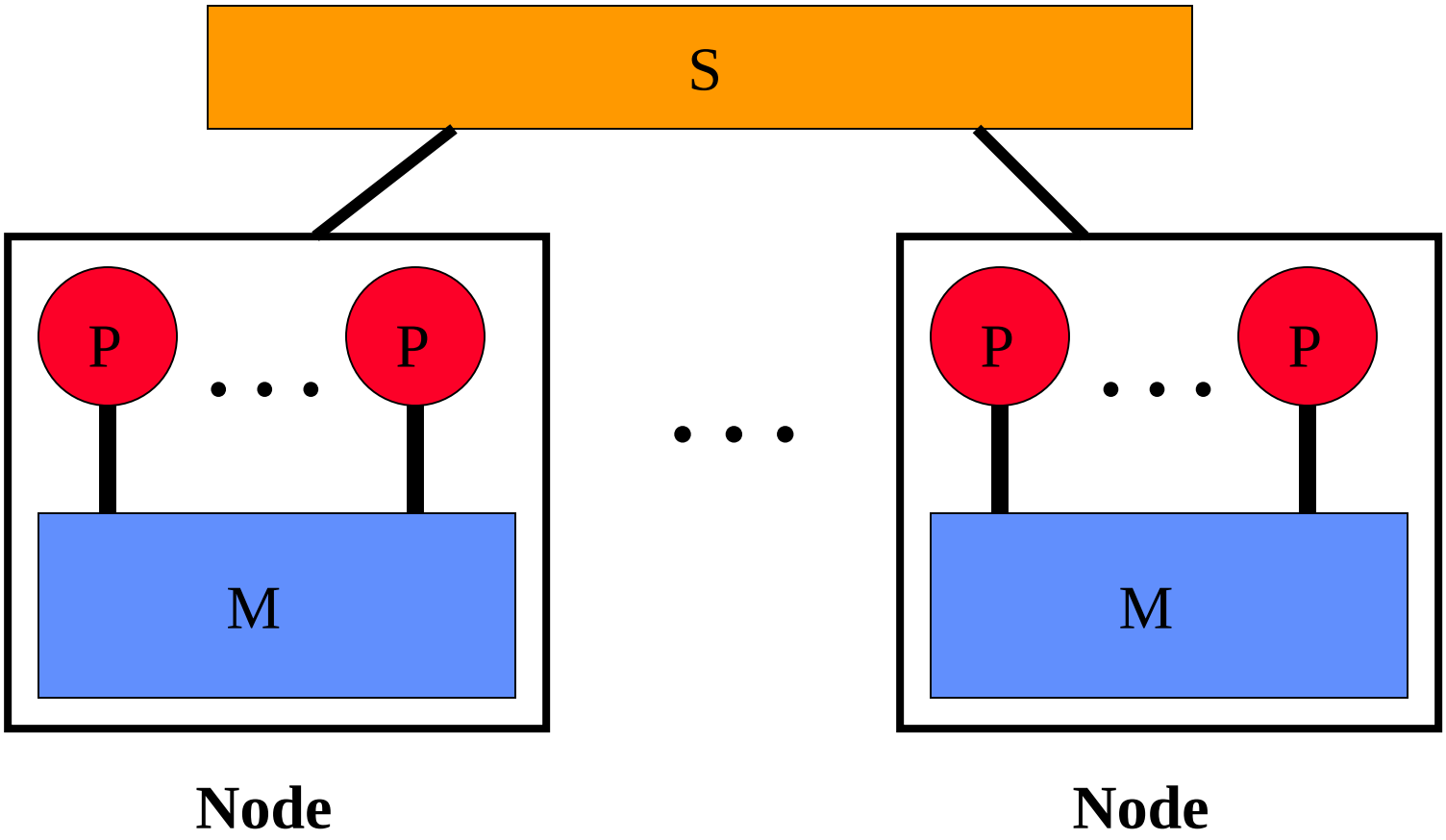
P=Processor
M=Memory
S=Switch




**Shared Memory** **Distributed Memory**

# IBM/CRAY Cluster
# (Distributed + Shared memory)

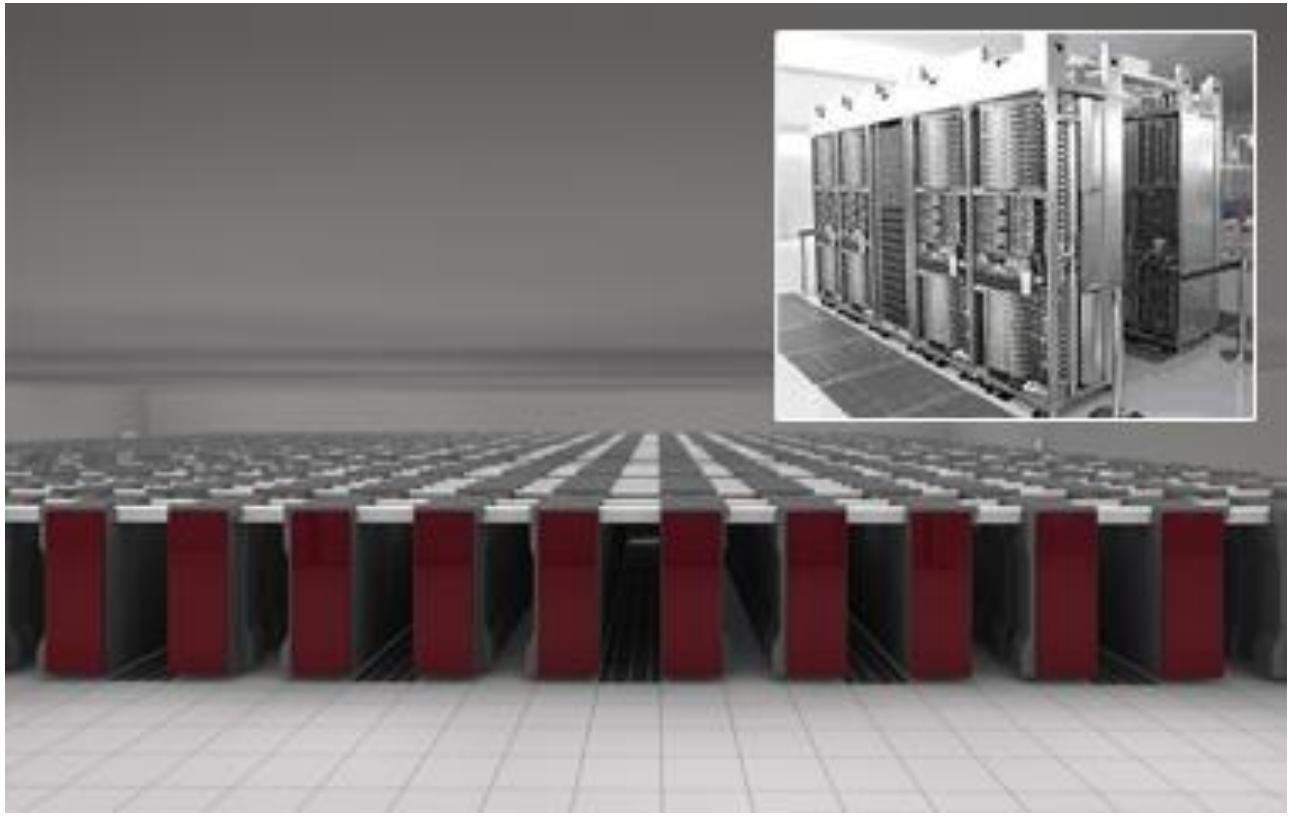
P=Processor
M=Memory
S=Switch



# CRAY XC-30 clusters at ECMWF

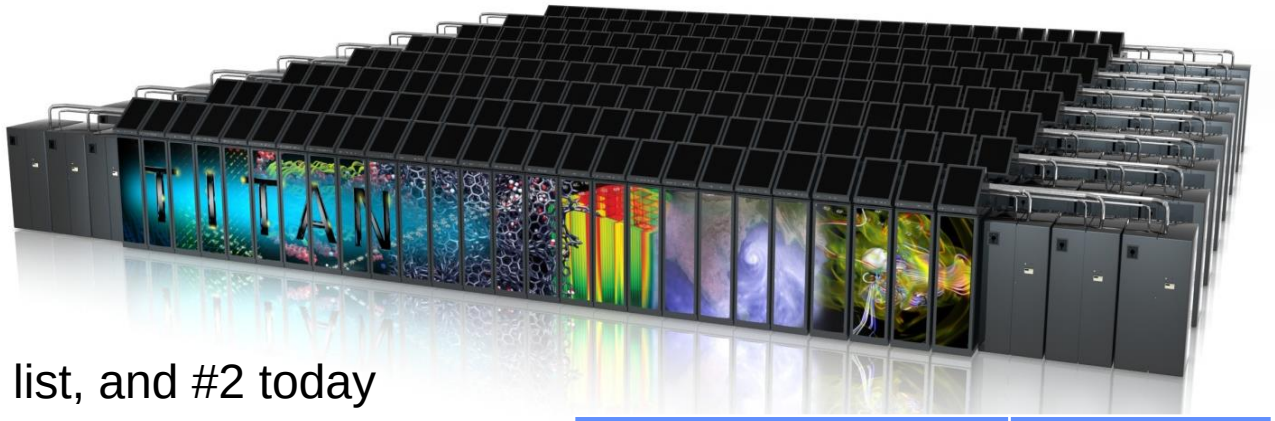
One of the TWO identical XC-30 clusters

# …and one the world's fastest (#4) and largest supercomputers – Fujitsu K computer



705,024 Sparc64 processor cores

# ORNL's "Titan" System



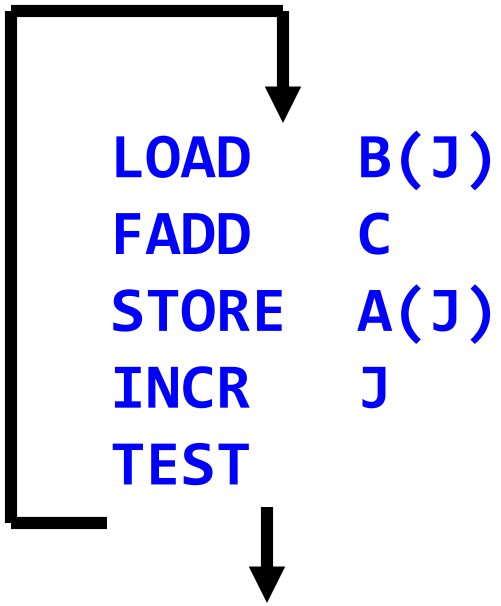

- #1 in Nov 2012 Top500 list, and #2 today
- 7.5X peak perf. of ECMWF's CRAY XC-30 clusters (CCA+CCB=3.6 Petaflops peak)
- Gemini interconnect
  - 3-D Torus
  - Globally addressable memory
- AMD Interlagos cores (16 cores per node)
- Accelerated node design using NVIDIA K20 "Kepler" GPUs
- 600 TB DDR3 mem. + 88 TB GDDR5 mem

| Titan Specs | |
|---|---|
| Compute Nodes | 18,688 |
| Login & I/O Nodes | 512 |
| Memory per node | 32 GB + 6 GB |
| # of NVIDIA K20 "Kepler" processors | 14,592 |
| Total System Memory | 688 TB |
| Total System Peak Performance | 27 Petaflops |

Source (edited): *James J. Hack, Director,* Oak Ridge National Laboratory

# Types of Processor

```
DO J=1,1000
  A(J)=B(J) + C
ENDDO
```




**SCALAR PROCESSOR**

Single instruction processes one element

```
LOADV   B->V1
FADDV   V1,C->V2
STOREV  V2->A
```

**VECTOR PROCESSOR**

Single instruction processes many elements

# The TOP500 project

- started in 1993
- Top 500 sites reported
- Report produced twice a year
  - EUROPE in JUNE/JULY (ISC16)
  - USA in NOV (SC16)
- Performance based on LINPACK benchmark
  - dominated by matrix multiply (DGEMM)
- High performance conjugate gradient (HPCG) benchmark announced at SC13
- http://www.top500.org/

# Top500: SC15 top 6 systems

| RANK | SITE | SYSTEM | CORES | RMAX (TFLOP/S) | RPEAK (TFLOP/S) | POWER (KW) |
|---|---|---|---|---|---|---|
| 1 | National Super Computer Center in Guangzhou<br>China | **Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P<br>NUDT | 3,120,000 | 33,862.7 | 54,902.4 | 17,808 |
| 2 | DOE/SC/Oak Ridge National Laboratory<br>United States | **Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x<br>Cray Inc. | 560,640 | 17,590.0 | 27,112.5 | 8,209 |
| 3 | DOE/NNSA/LLNL<br>United States | **Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom<br>IBM | 1,572,864 | 17,173.2 | 20,132.7 | 7,890 |
| 4 | RIKEN Advanced Institute for Computational Science (AICS)<br>Japan | K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect<br>Fujitsu | 705,024 | 10,510.0 | 11,280.4 | 12,660 |
| 5 | DOE/SC/Argonne National Laboratory<br>United States | **Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom<br>IBM | 786,432 | 8,586.6 | 10,066.3 | 3,945 |
| 6 | DOE/NNSA/LANL/SNL<br>United States | **Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect<br>Cray Inc. | 301,056 | 8,100.9 | 11,078.9 | |

# ECMWF in Top 500 (SC15)

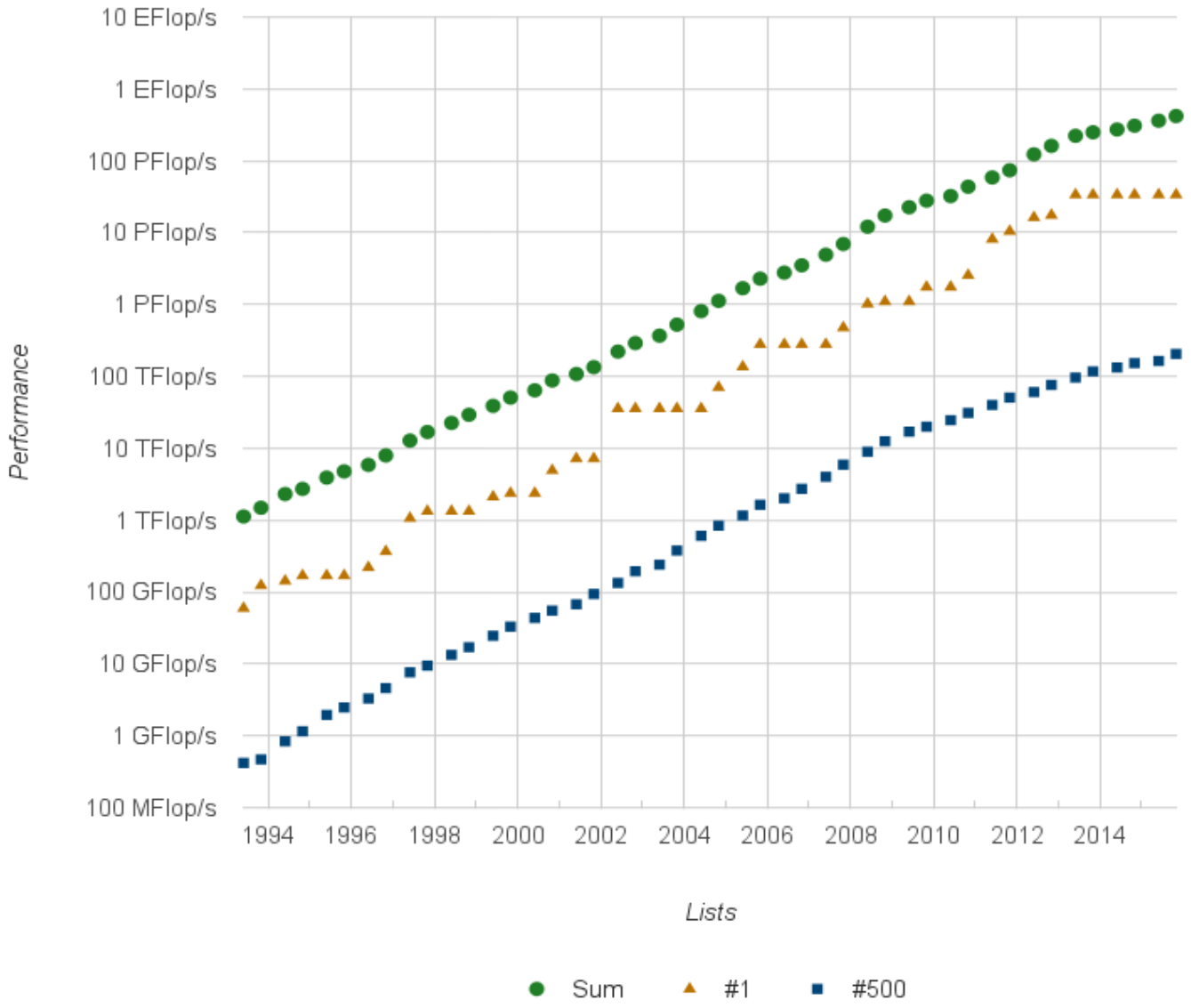
TFlops

| Rank | Site | Computer/Year Vendor | Cores | $R_{max}$ | $R_{peak}$ | Power |
|---|---|---|---|---|---|---|
| 46 | ECMWF<br>United Kingdom | Cray XC30, Intel Xeon E5-2697v2 12C 2.7GHz, Aries interconnect<br>Cray Inc. | 83,160 | 1,552.0 | 1,796.3 | |
| 47 | ECMWF<br>United Kingdom | Cray XC30, Intel Xeon E5-2697v2 12C 2.7GHz, Aries interconnect<br>Cray Inc. | 83,160 | 1,552.0 | 1,796.3 | |

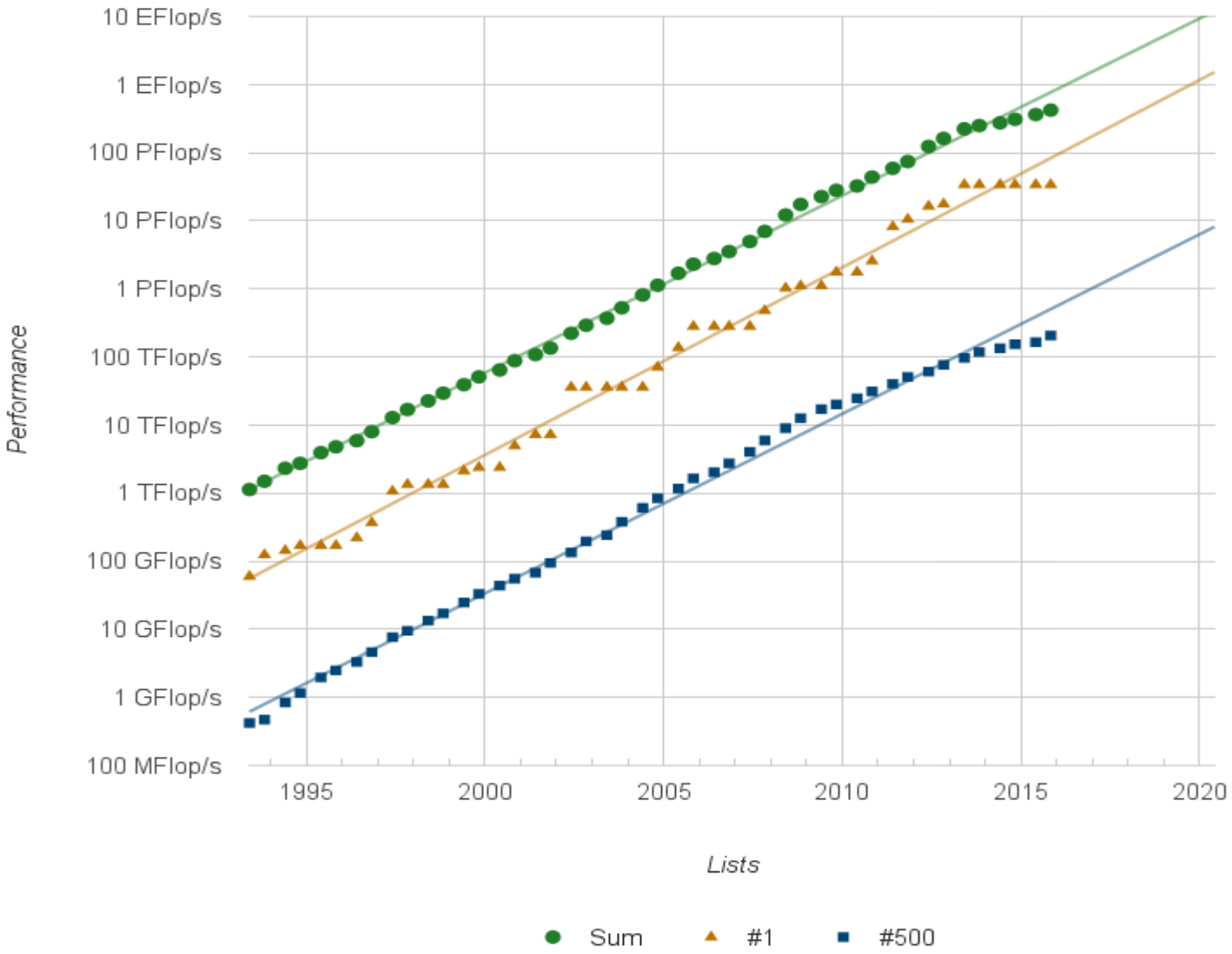
$R_{max}$ – Tflop/sec achieved with LINPACK Benchmark

$R_{peak}$ – Peak Hardware Tflop/sec (that will never be reached!)

# Top500: Performance Development



# Top500: Projected Performance Development



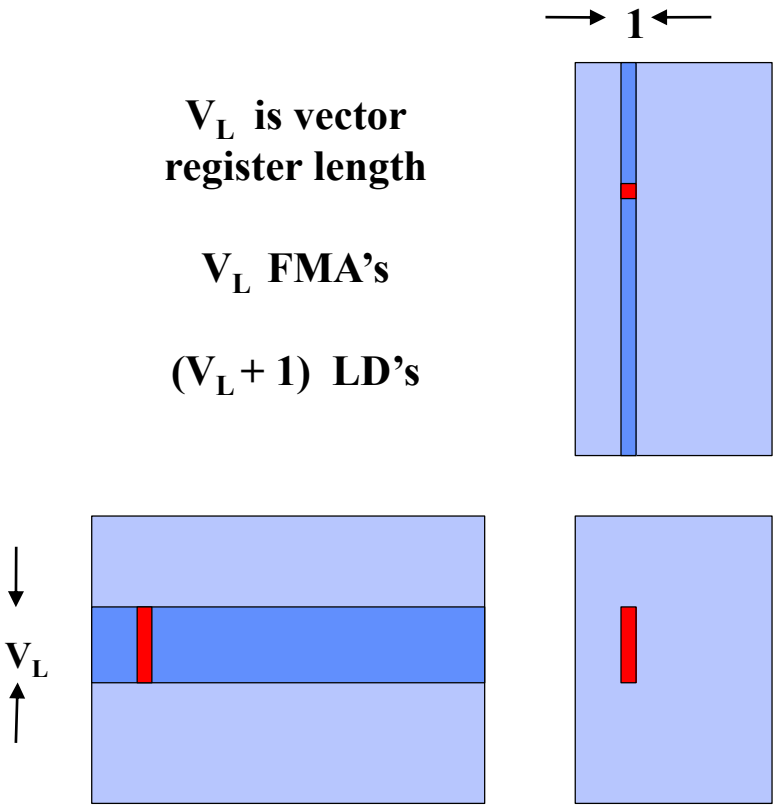
# Why is Matrix-Matrix Multiply (DGEMM) so efficient?

## VECTOR

$V_L$ is vector register length

$V_L$ FMA's

$(V_L + 1)$ LD's



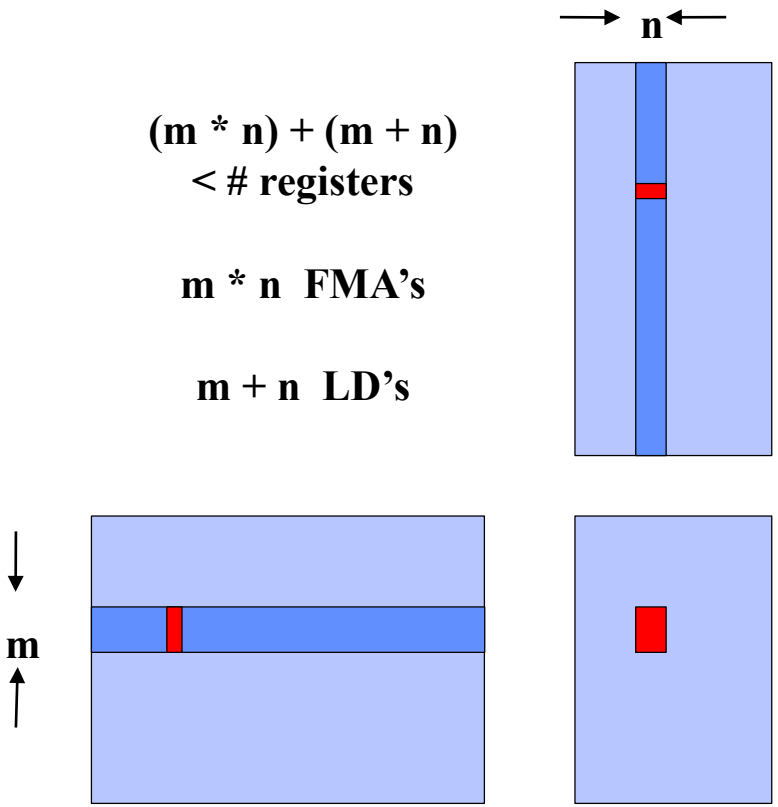

FMA's ~= LD's

## SCALAR / CACHE

(m * n) + (m + n) < # registers

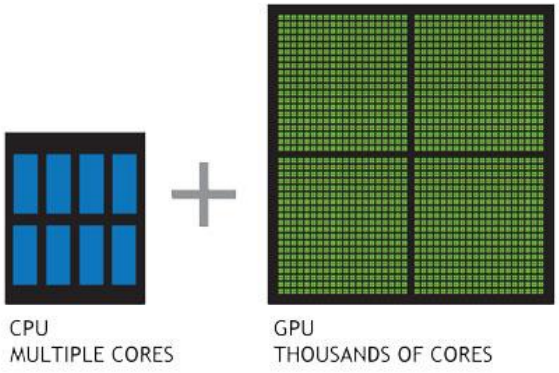
m * n FMA's

m + n LD's




FMA's >> LD's

# Accelerators (2 main types)

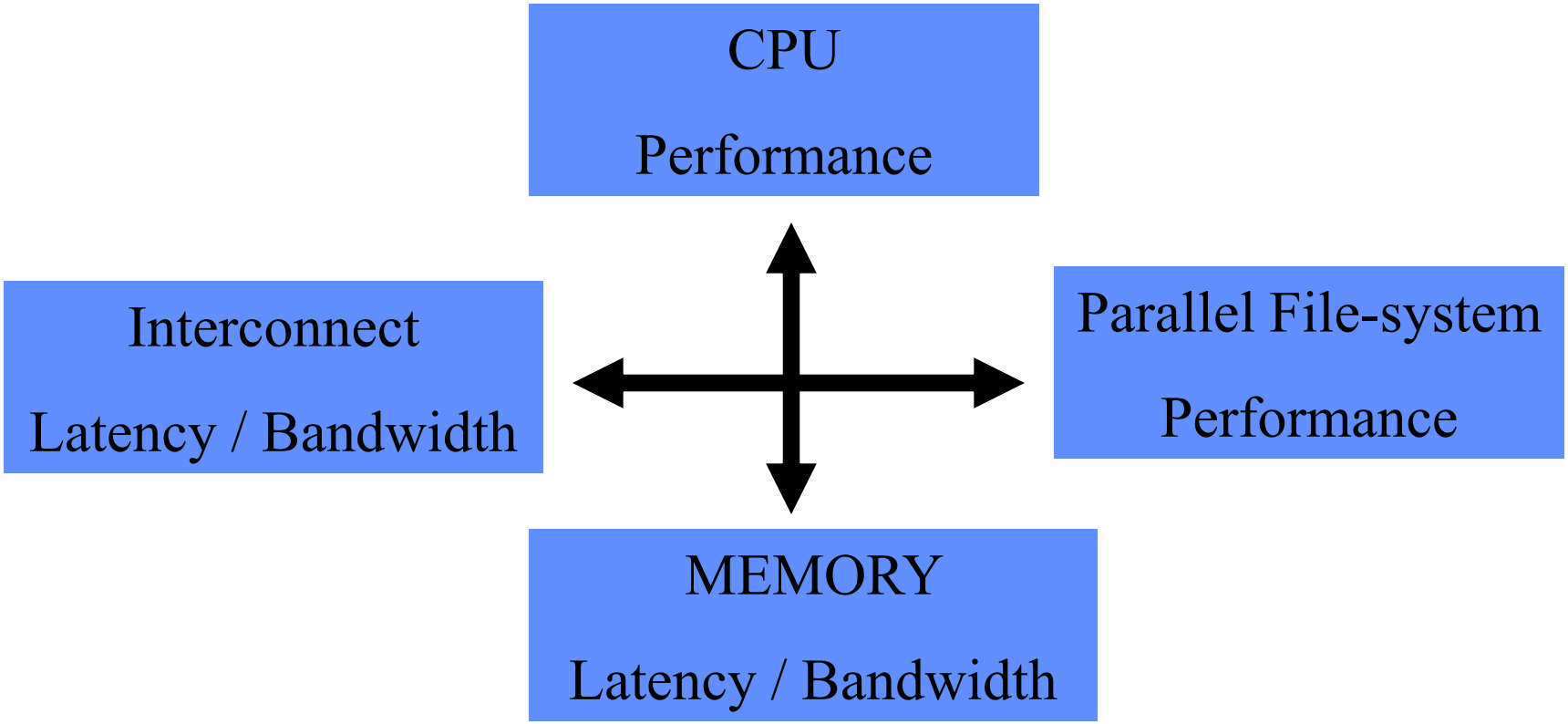



- **GPU – Graphics Processing Unit**
  - **High performance, low power, but 'challenging' to program for large applications, separate memory, GPU/CPU interface (PCIx up to 16GB/sec today, in future NVLINK up to 80GB/sec between GPUs in same node)**
  - **Expect GPU technology to be more easily useable on future HPCs**
  - **http://gpgpu.org/developer**
  - **GPU hardware today mainly supplied by NVIDIA**
- **INTEL (Xeon Phi, aka "MIC")**
  - **"Knights Corner" from 2012 requires CPU host (via PCIx)**
  - **"Knights Landing" from 2016, does not require CPU host (64-72 cores)**
  - **"Knights Hill" from 2018, does not require CPU host**

# Pre-Exascale US Department of Energy Supercomputers

| | TITAN | CORI (phase2) | SUMMIT | AURORA |
|---|---|---|---|---|
| Available | 2012 | mid-2016 | 2017 | 2018 |
| Laboratory | Oak Ridge | NERSC | Oak Ridge | Argonne |
| Peak Performance | 27 PF | 28 PF | 150 PF | 180 PF |
| Power | 9 MW | < 3.7 MW | 10 MW | 13 MW |
| Processors | AMD Opteron + NVIDIA Kepler K20X GPU via PCI Gen2 | INTEL Xeon Phi (Knights Landing) | IBM Power 9 + NVIDIA Volta GPU via NVLINK | INTEL Xeon Phi (Knights Hill) |
| Vendor | CRAY | CRAY | IBM | CRAY |

# Key Architectural Features of a Supercomputer




"a balancing act to achieve good sustained performance"

# Challenges in parallel computing

- **Parallel Computers**
  - **Have ever increasing processors, memory, performance, but**
  - **Need more space (new computer halls = $)**
  - **Need more power (MWs = $)**
- **Parallel computers require/produce a lot of data (I/O)**
  - **Require parallel file systems (GPFS, Lustre) + archive store**
- **Applications need to scale to increasing numbers of processors, problems areas are**
  - **Load imbalance, Serial sections, Global Communications**
  - **Increasing resolution => shorter time-steps => more time-steps**
- **Debugging parallel applications (totalview, ddt)**
- **We are going to be using more processors in the future!**
- **More cores per socket, little/no clock speed improvements**

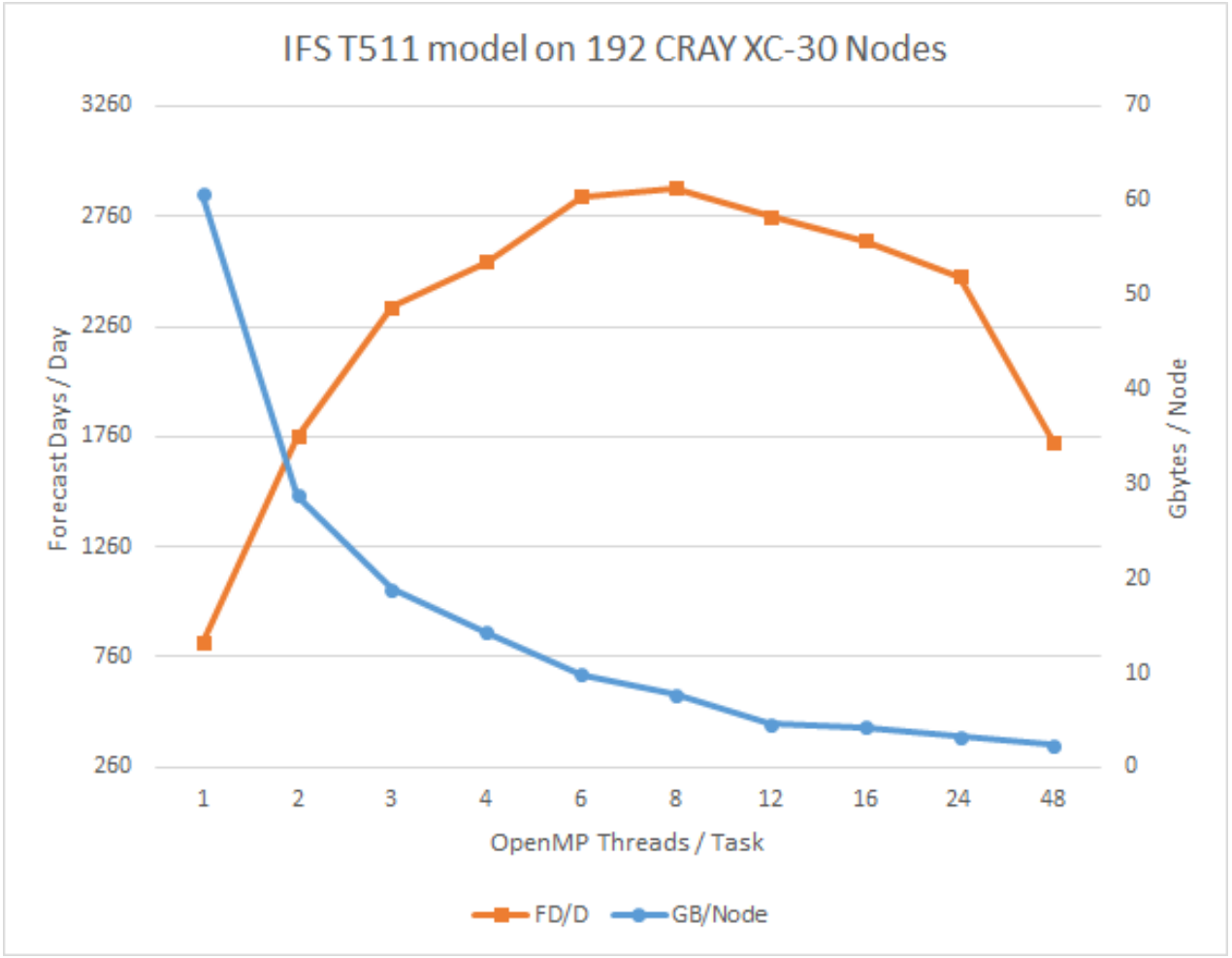
# Parallel Programming Languages

- **OpenMP**
  - **directive based (www.openmp.org)**
  - **support for Fortran and C/C++**
  - **shared memory programming only**
- **OpenACC**
  - **directive based (www.openacc.org)**
  - **support for Fortran and C**
  - **GPU programming (e.g. NVIDIA)**
- **PGAS (Partitioned Global Address Space)**
  - **UPC, Fortran 2008 Coarrays**
  - **One programming model for inter and intra node parallelism**
  - **One-sided communication**

# OpenMP example

```
!$OMP PARALLEL DO SCHEDULE(STATIC,1)&
!$OMP& PRIVATE(JMLOCF,IM,ISTA,IEND)
      DO JMLOCF=NPTRMF(MYSETN),NPTRMF(MYSETN+1)-1
        IM=MYMS(JMLOCF)
        ISTA=NSPSTAF(IM)
        IEND=ISTA+2*(NSMAX+1-IM)-1
        CALL SPCSI(CDCONF,IM,ISTA,IEND,LLONEM,ISPEC2V,&
         &ZSPVORG,ZSPDIVG,ZSPTG,ZSPSPG)
      ENDDO
!$OMP END PARALLEL DO
```

# Why OpenMP? Ans: For performance and memory




Testing combinations 9216Tx1t, 4608Tx2t, 3072Tx3t, 1536Tx6t, 768Tx12t, 384Tx24t and 192Tx48t

# OpenACC example

```
!$acc parallel loop copyin(dt,rmass), &
!$acc private(i,j), present(pos,vel,f,a,np,nd)
do i = 1,np
  do j = 1,nd
    pos(j,i) = pos(j,i) + vel(j,i)*dt + 0.5*dt*dt*a(j,i)
    vel(j,i) = vel(j,i) + 0.5*dt*(f(j,i)*rmass + a(j,i))
    a(j,i) = f(j,i)*rmass
  enddo
enddo
!$acc end parallel loop
```

http://www.ecmwf.int/sites/default/files/elibrary/2014/13673-challenges-getting-ecmwfs-weather-forecast-model-ifs-exascale.pdf
Link includes results of a port of IFS spectral transform kernel to GPU using OpenACC

# Fortran2008 coarray (PGAS) example

```
!$OMP PARALLEL DO SCHEDULE(DYNAMIC,1) PRIVATE(JM,IM,JW,IPE,ILEN,ILENS,IOFFS,IOFFR)
DO JM=1,D%NUMP
  IM = D%MYMS(JM)
  CALL LTINV(IM,JM,KF_OUT_LT,KF_UV,KF_SCALARS,KF_SCDERS,ILEI2,IDIM1,&
    & PSPVOR,PSPDIV,PSPSCALAR ,&
    & PSPSC3A,PSPSC3B,PSPSC2 , &
    & KFLDPTRUV,KFLDPTRSC,FSPGL_PROC)
  DO JW=1,NPRTRW
    CALL SET2PE(IPE,0,0,JW,MYSETV)
    ILEN = D%NLEN_M(JW,1,JM)*IFIELD
    IF( ILEN > 0 )THEN
      IOFFS = (D%NSTAGT0B(JW)+D%NOFF_M(JW,1,JM))*IFIELD
      IOFFR = (D%NSTAGT0BW(JW,MYSETW)+D%NOFF_M(JW,1,JM))*IFIELD
      FOUBUF_C(IOFFR+1:IOFFR+ILEN)[IPE]=FOUBUF_IN(IOFFS+1:IOFFS+ILEN)
    ENDIF
    ILENS = D%NLEN_M(JW,2,JM)*IFIELD
    IF( ILENS > 0 )THEN
      IOFFS = (D%NSTAGT0B(JW)+D%NOFF_M(JW,2,JM))*IFIELD
      IOFFR = (D%NSTAGT0BW(JW,MYSETW)+D%NOFF_M(JW,2,JM))*IFIELD
      FOUBUF_C(IOFFR+1:IOFFR+ILENS)[IPE]=FOUBUF_IN(IOFFS+1:IOFFS+ILENS)
    ENDIF
  ENDDO
ENDDO
!$OMP END PARALLEL DO
SYNC IMAGES(D%NMYSETW)
FOUBUF(1:IBLEN)=FOUBUF_C(1:IBLEN)[MYPROC]
```

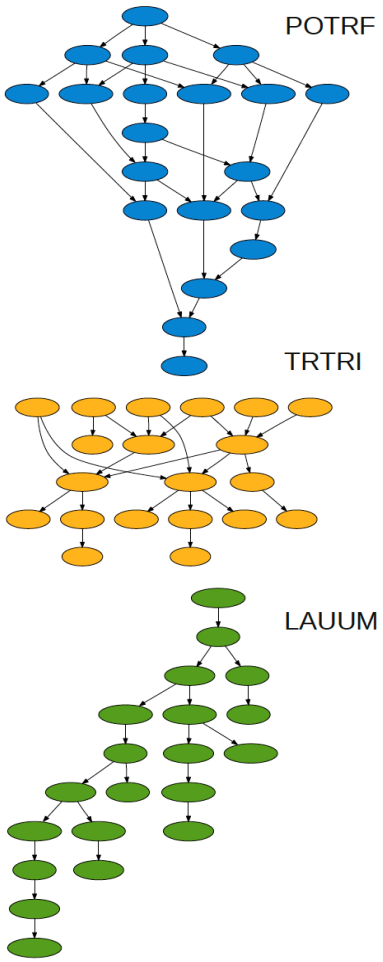
# Parallel Programming Libraries

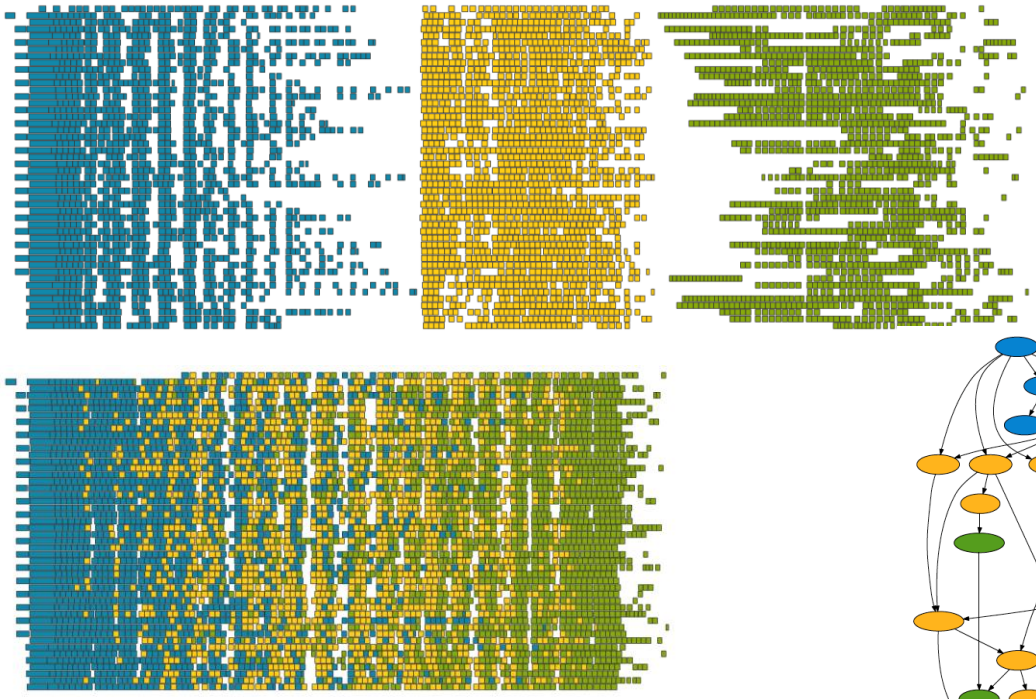
- **MPI**
  - **Most widely used since mid-90's (www.mpi-forum.org)**
  - **MPI-3.0 standard is 852 pages!**
  - **MPI-2.2 is the default MPI on most systems**
  - **Most users will use a small subset of MPI facilities**
  - **Use collectives (e.g. MPI_alltoallv) and non-blocking calls for performance**
  - **MPI-only application scaling issues?**
- **GASPI/GPI**
  - **PGAS one-sided programming (www.gpi-site.com/gpi2)**
  - **Interoperable with MPI**
  - **Comparable 'notify' technology expected to be in MPI-4**
  - **Supported in experimental version of IFS (as an alternative to coarrays)**

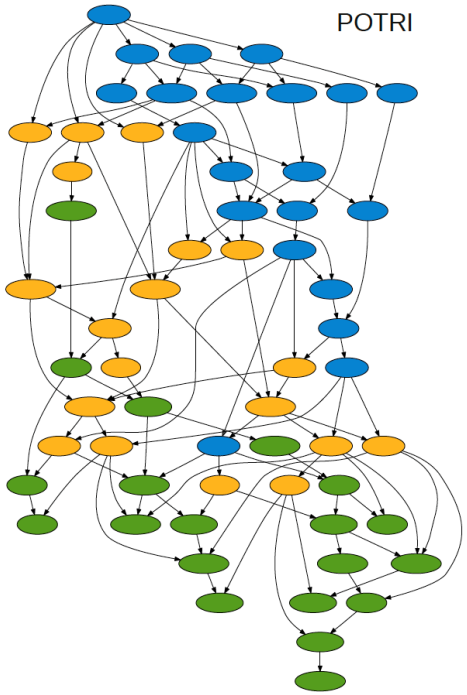
# Parallel Programmers use…

- **Fortran, C/C++ with MPI for communicating between tasks**
  - **works for applications running on shared and distributed memory systems**
- **Fortran, C/C++ with OpenMP**
  - **For applications that need performance that is satisfied by a single node (shared memory)**
- **Hybrid combination of MPI/OpenMP**
  - **ECMWF's IFS uses this approach (over 15 years now)**
- **Hybrid combination of MPI/OpenACC (for GPU)**
  - **Meteo-Swiss have ported COSMO to NVIDIA GPU**
- **Early years for DAGs (e.g. MPI + OmpSs)**

# DAG example: Cholesky Inversion






DAG = Directed Acyclic Graph

Can IFS use this technology?

Source: Stan Tomov, ICL, University of Tennessee, Knoxville

# Topics in Parallel Computing …

Cache, Cache line

Domain decomposition

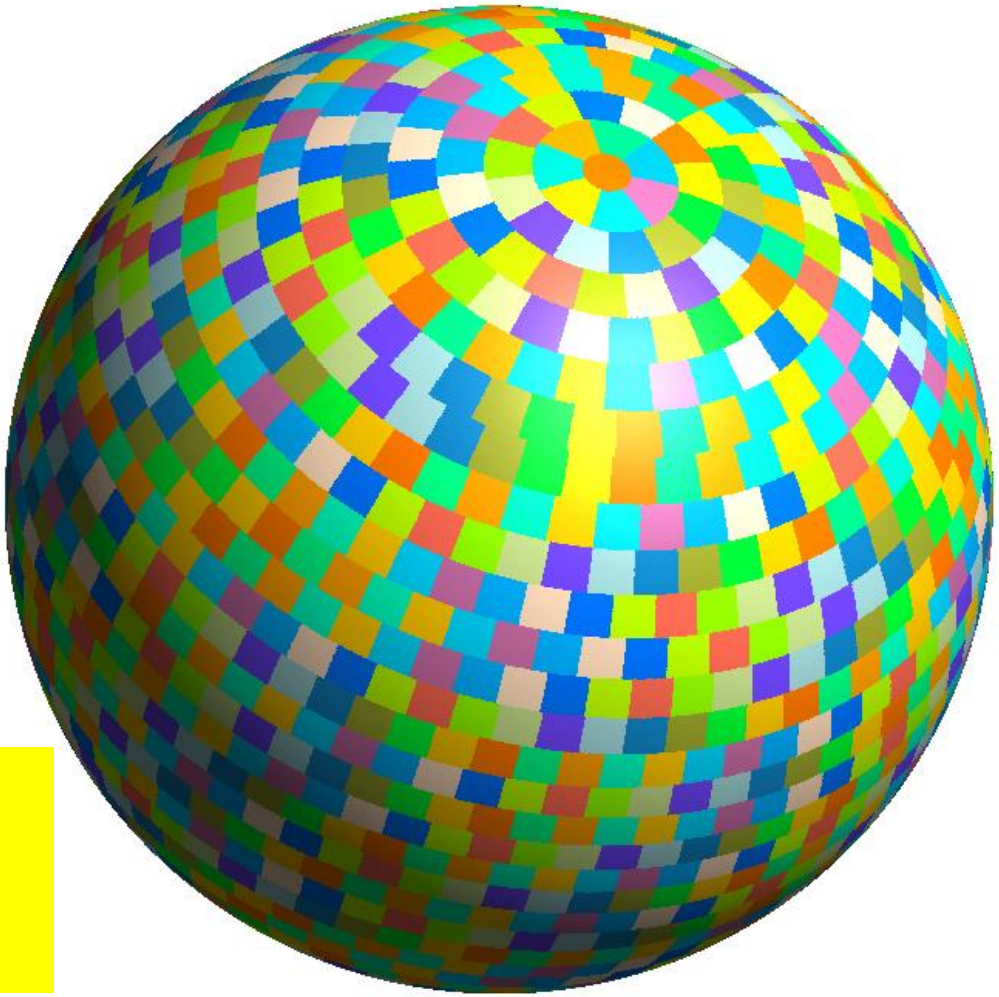
Halo, halo exchange

Load imbalance

Synchronization

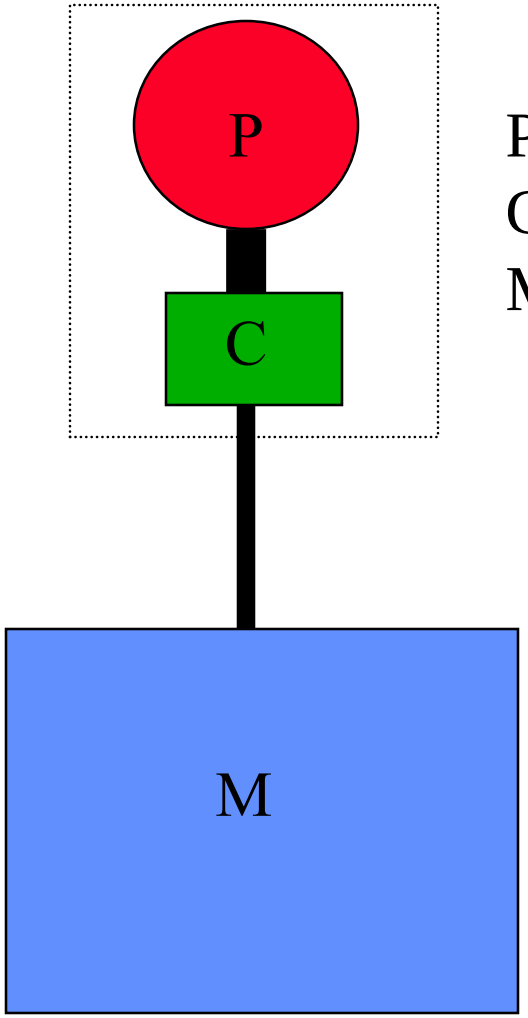
Barrier

IFS domain decomposition for 1,600 MPI tasks used in TCo1279 operational model today. We have run an IFS model on TITAN with up to 28,672 MPI tasks each with 8 cores (OpenMP threads), for a total of 229,376 cores.

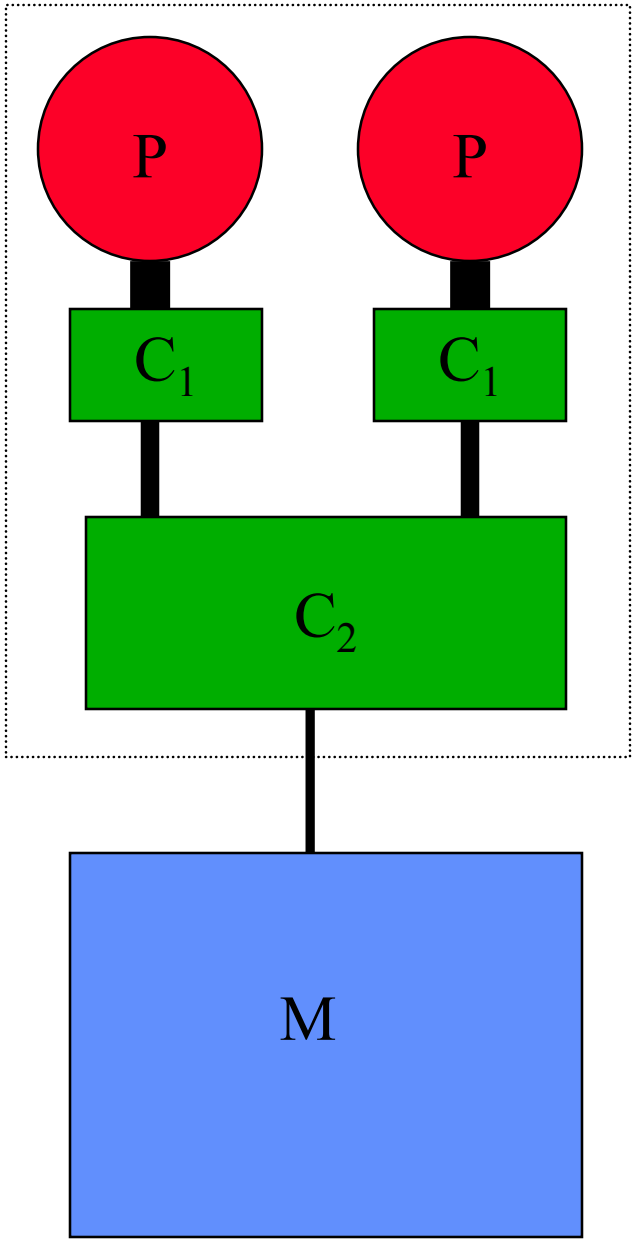


Thank you to Willem Deconinck for providing this graphic.

# Cache




P=Processor
C=Cache
M=Memory



# Cache on scalar systems

- **Processors are 100's of cycles away from Memory**
- **Cache is a small (and fast) memory closer to processor**
- **Cache line typically 128 bytes**
- **Good for cache performance**
  - **Single stride access is always the best**
  - **Over inner loop leftmost index (fortran)**

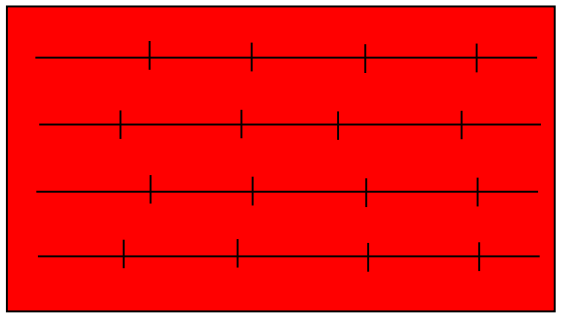
**BETTER**

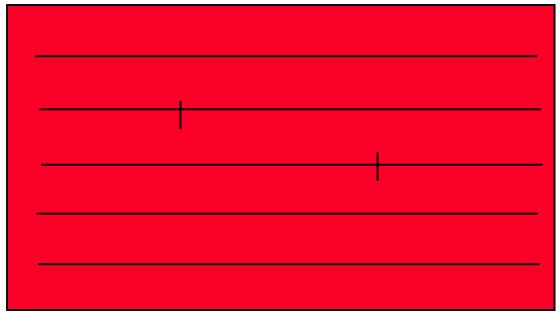
```
DO J=1,N
  DO I=1,M
    A(I,J)= . . .
  ENDDO
ENDDO
```

**WORSE**

```
DO J=1,N
  DO I=1,M
    A(J,I)= . . .
  ENDDO
ENDDO
```

# IFS Grid-Point Calculations (cache blocking example)



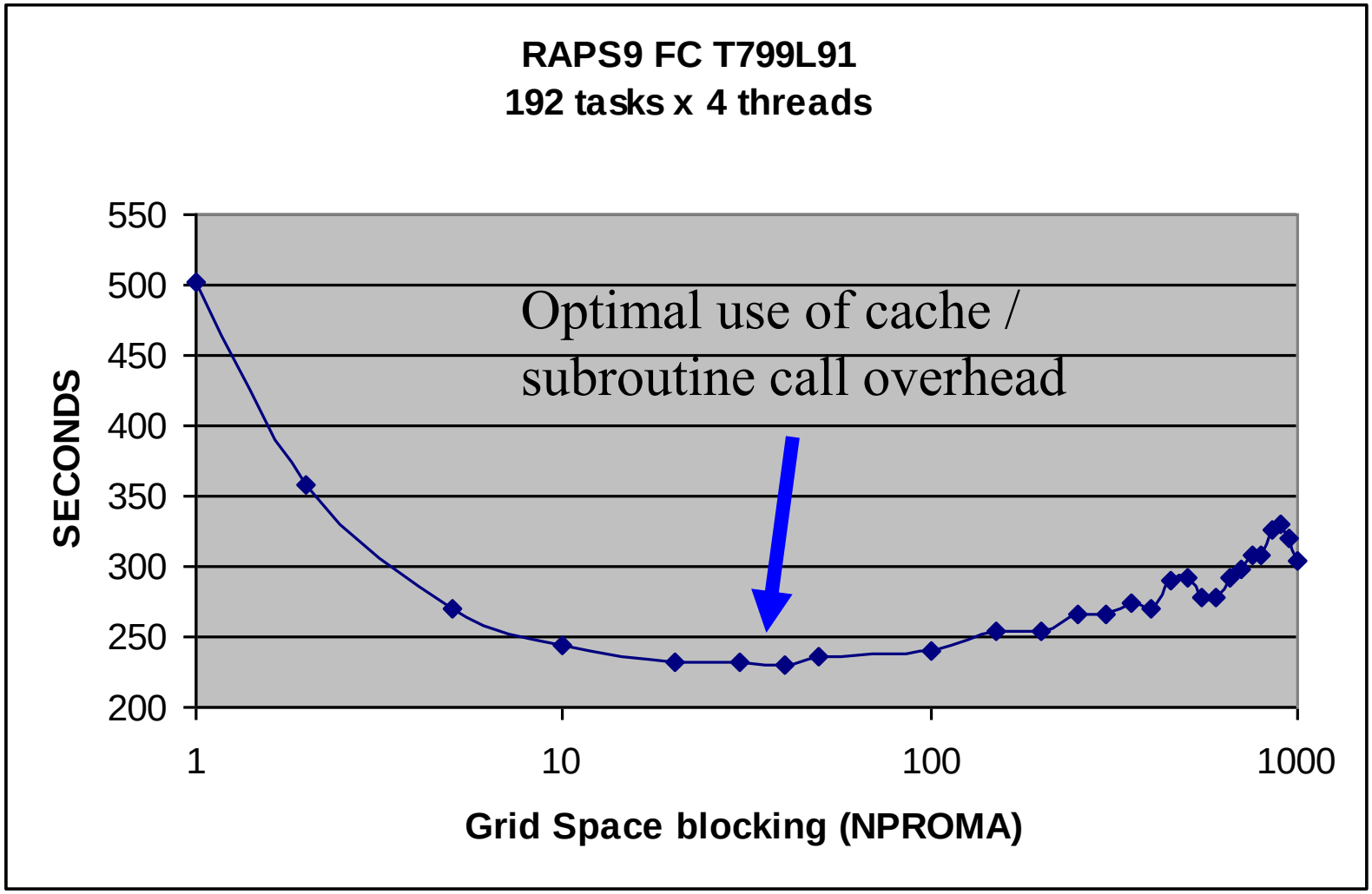

```
U(NGPTOT,NLEV)

NGPTOT = NLAT * NLON
NLEV = vertical levels
```

```
DO J=1, NGPTOT, NPROMA
  CALL GP_CALCS
ENDDO
```

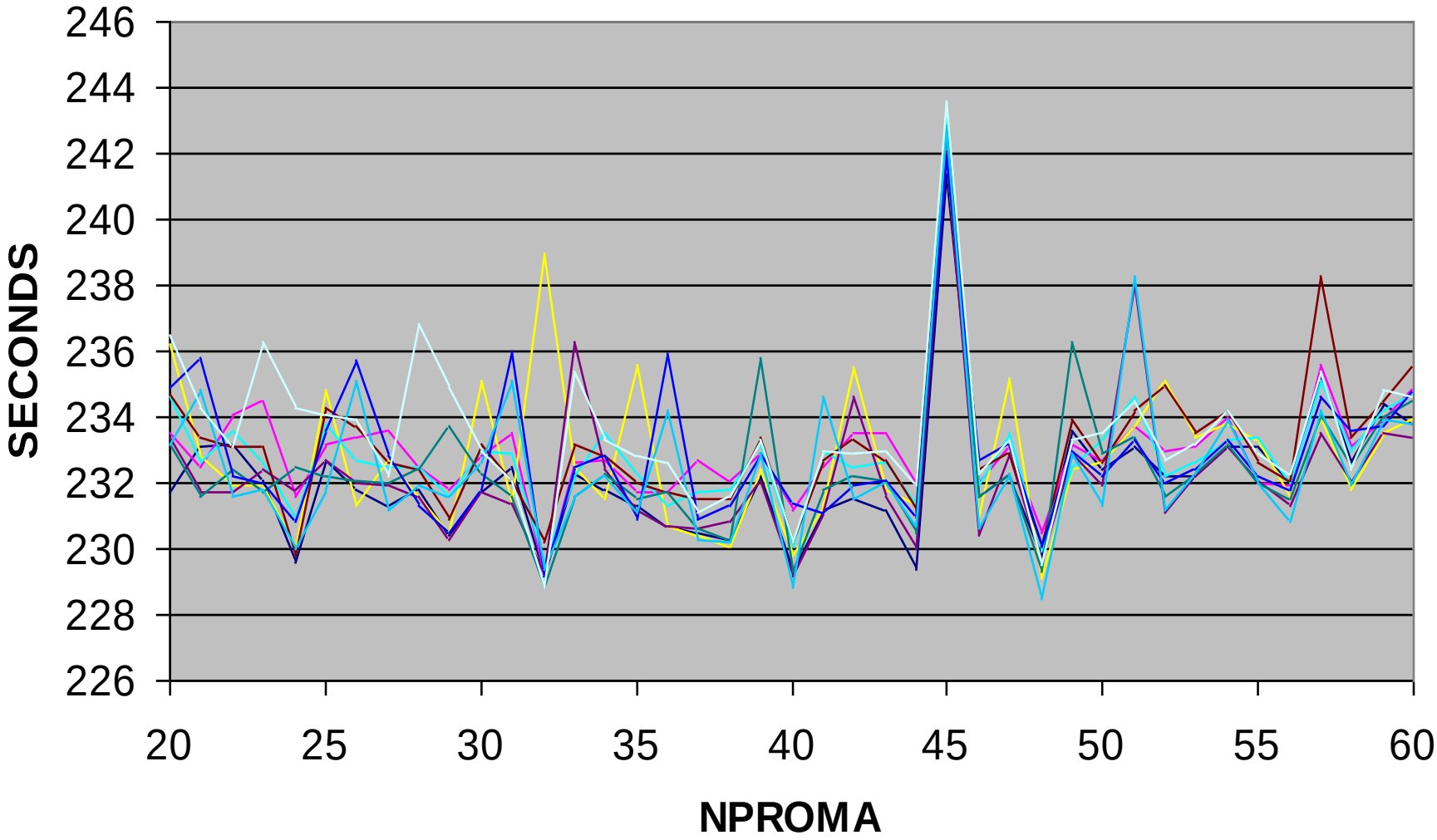
Lots of work
Independent for each J

```
SUB GP_CALCS

DO I=1,NPROMA
ENDDO

END
```

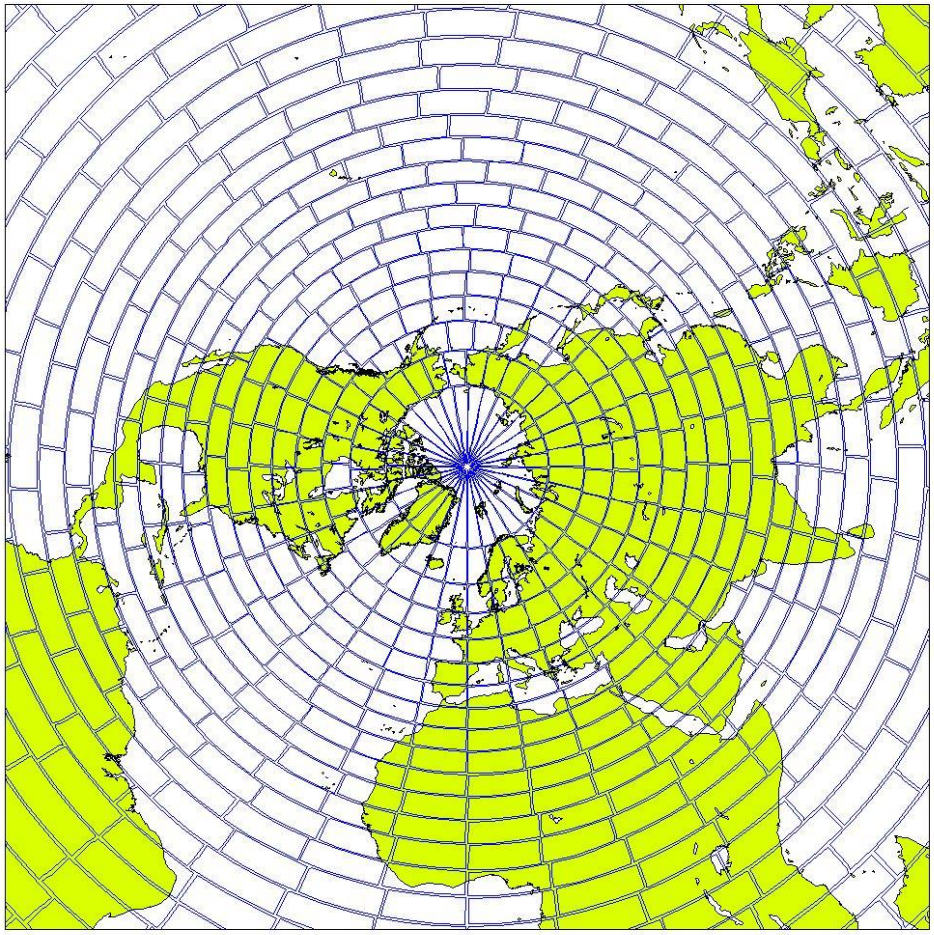
# Grid point space blocking for Cache



# T799 FC 192x4 (10 runs)

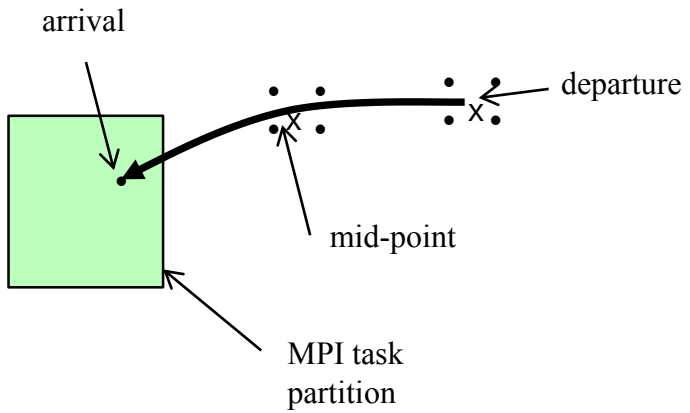


# $T_L$799 1024 tasks 2D partitioning (used in past)



2D partitioning results in non-optimal Semi-Lagrangian comms requirement at poles and equator!

Square shaped partitions are better than rectangular shaped partitions.



# eq_regions partitioning algorithm (used in IFS)



FIG. 1.1. *Partition* EQ(2, 33)

*where* $e(x, y)$ *is the* $\mathbb{R}^{d+1}$ *Euclidean distance* $\|x - y\|$.

The following definitions are specific to the main theorems stated in this paper.

DEFINITION 1.3. *A set* $Z$ *of partitions of* $\mathbb{S}^d$ *is said to be* diameter-bounded *with* diameter bound $K \in \mathbb{R}_+$ *if for all* $P \in Z$, *for each* $R \in P$,

$$\operatorname{diam} R \leqslant K |P|^{-1/d}.$$

DEFINITION 1.4. *The set of recursive zonal equal area partitions of* $\mathbb{S}^d$ *is defined as*

$$\mathrm{EQ}(d) := \{\mathrm{EQ}(d, N) \mid N \in \mathbb{N}_+\}. \tag{1.2}$$

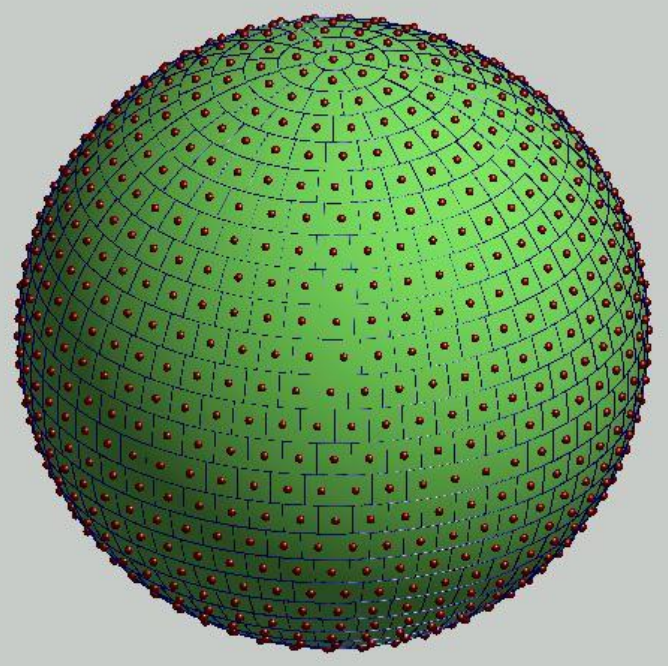
*where* $\mathrm{EQ}(d, N)$ *denotes the recursive zonal equal area partition of the unit sphere* $\mathbb{S}^d$ *into* $N$ *regions, which is defined via the algorithm given in Section 3.*

This paper claims that the partition defined via the algorithm given in Section 3 is an equal area partition which is diameter bounded. This is formally stated in the following theorems.

THEOREM 1.5. *For* $d \geqslant 1$ *and* $N \geqslant 1$, *the partition* $\mathrm{EQ}(d, N)$ *is an equal area partition of* $\mathbb{S}^d$.

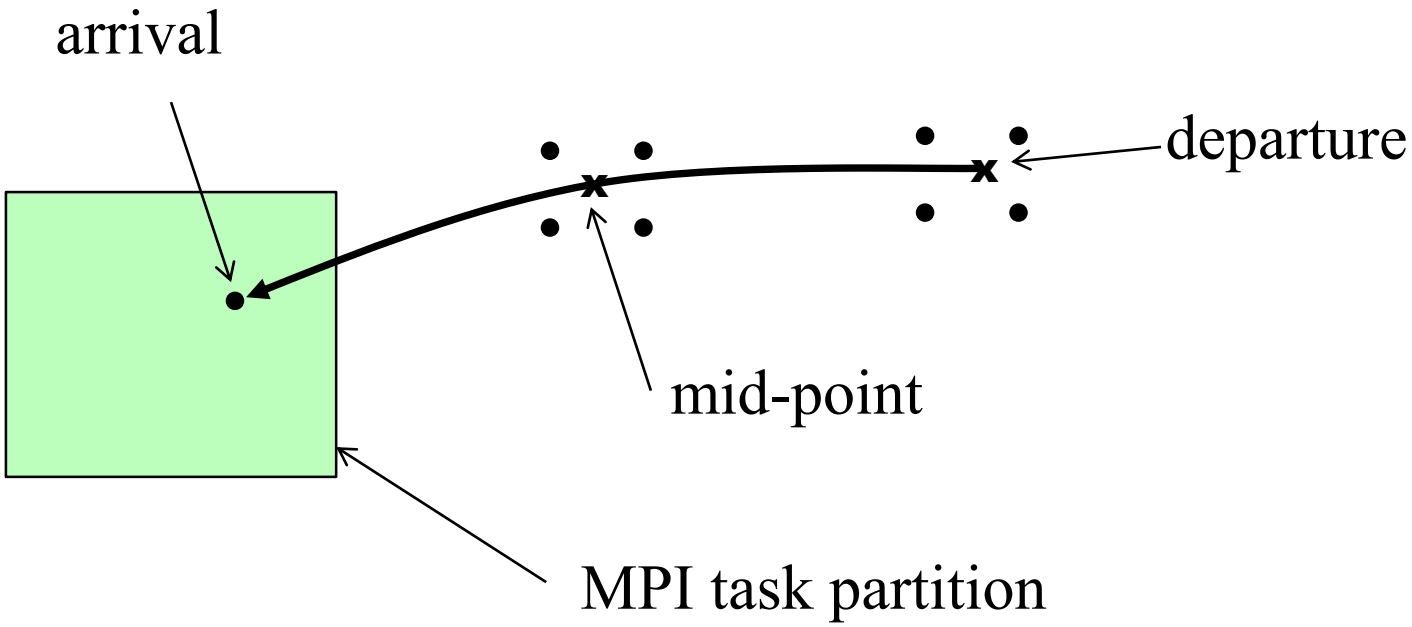
THEOREM 1.6. *For* $d \geqslant 1$, $\mathrm{EQ}(d)$ *is diameter-bounded in the sense of Definition 1.3.*

**Recursive zonal equal area partition of $S^2$ into 1024 regions, showing the center point of each region.**

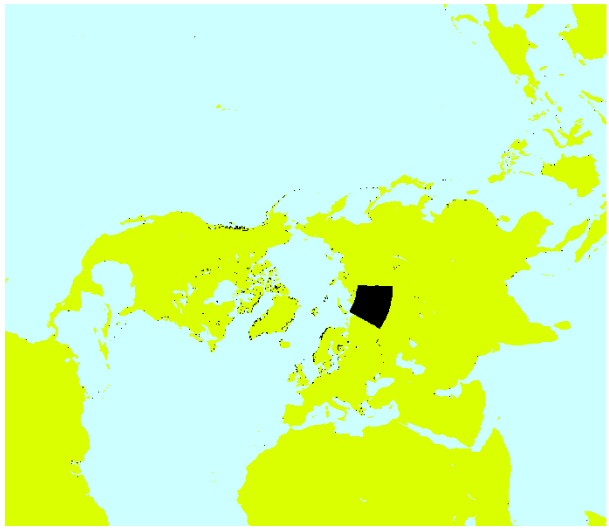


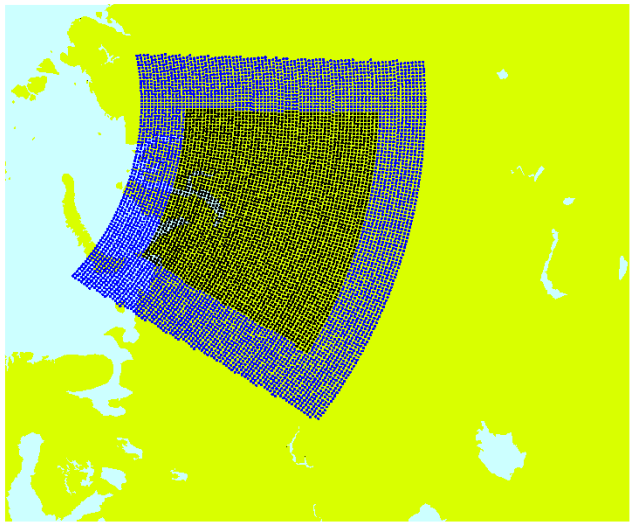
# Halo example : IFS Semi-Lagrangian Transport

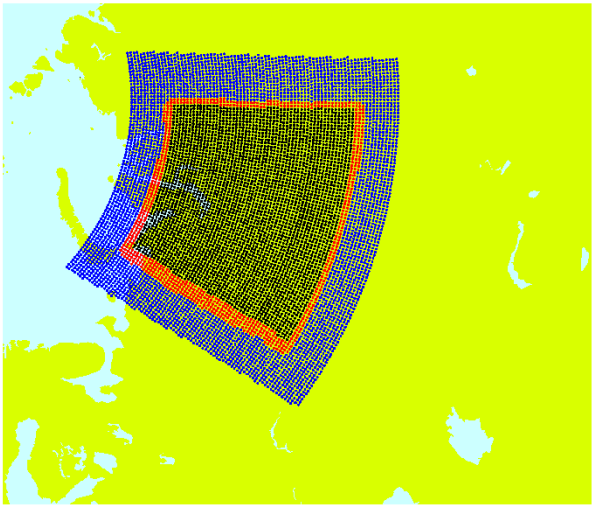
- **Computation of a trajectory from each grid-point backwards in time, and**
- **Interpolation of various quantities at the departure and at the mid-point of the trajectory**

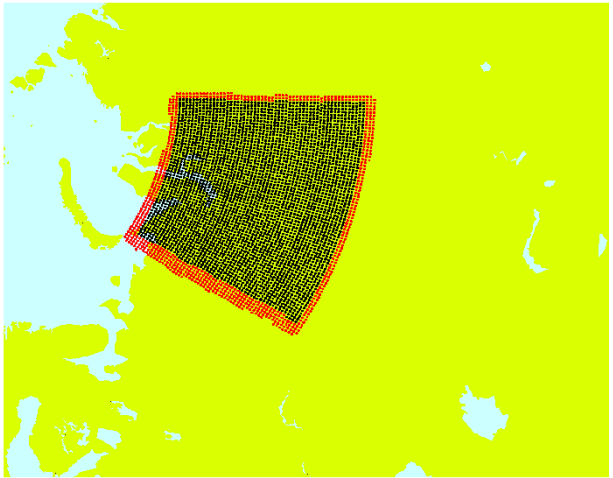


# Halo's in IFS (T799 model, 256 tasks, showing task 11)









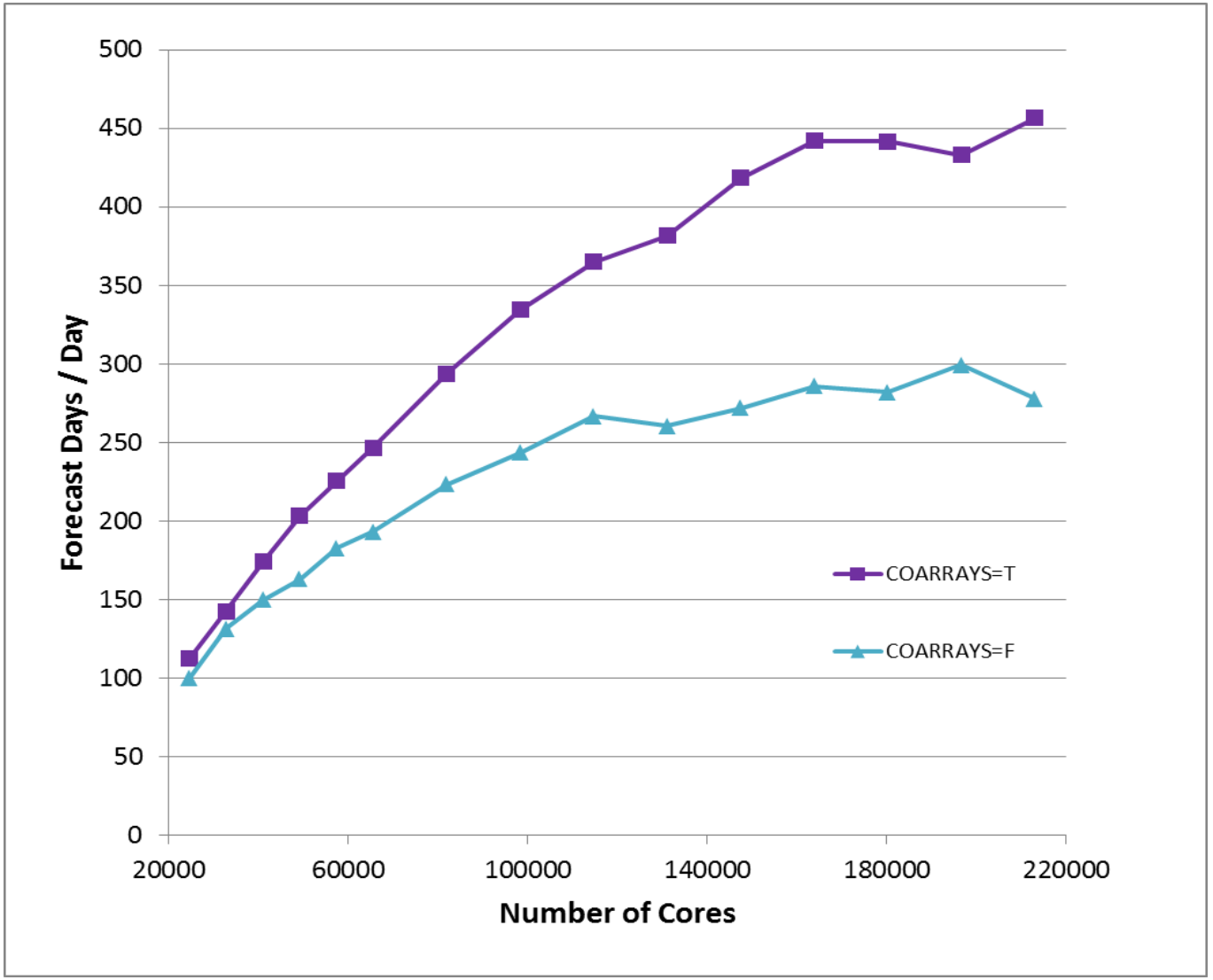
Black – grid points owned by task 11

Blue – halo grid points , max wind x time-step

Red – grid points in halo actually used by task 11

Bottom two graphics
LH – using MPI
RH – using Fortran2008 coarrays (PGAS)

# 5 km IFS model scaling on TITAN (with and without Fortran2008 coarrays)



# Characteristics of codes that will perform well on all parallel computers

- **Computation**
  - **High computational intensity (flops/data words accessed)**
  - **Little use of memory bandwidth**
- **Memory**
  - **Locality of reference**
  - **Registers or first level cache**
- **Communication**
  - **Infrequent nearest neighbour communication**
- **Input/Output**
  - **Relatively low volume, or**
  - **Parallel implementation (in dedicated nodes)**


Thank you for your attention.
Questions?