The Road to Exascale and Legacy Software for Dense Linear Algebra

Jack Dongarra
University of Tennessee &
Oak Ridge National Lab

The Road to Exascale and Legacy Software for Dense Linear Algebra
(or What I've Been Doing for the Last 43 Years)

Jack Dongarra
University of Tennessee &
Oak Ridge National Lab

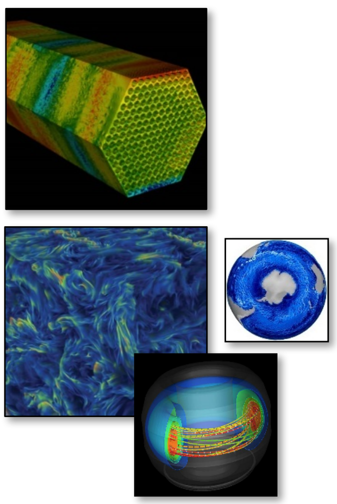
Outline for the Talk

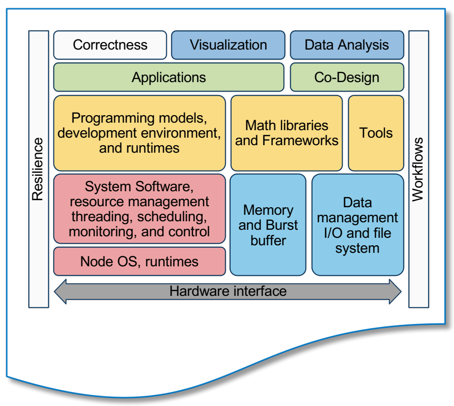
- What was going on before
- What’s the current situation
- What’s planned for exascale

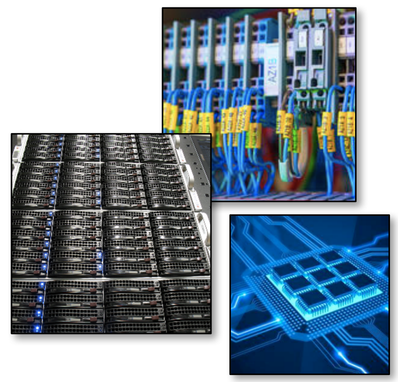
But first, a word about the DOE Exascale Computing Program

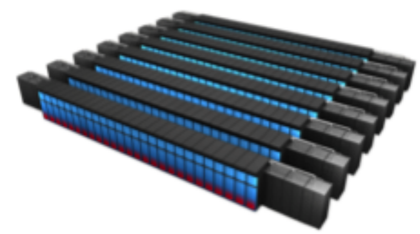
DOE ECP has formulated a holistic approach that uses co-design and integration to achieve capable exascale

Application Development	Software Technology	Hardware Technology	Exascale Systems
Science and mission applications	Scalable and productive software stack	Hardware technology elements	Integrated exascale supercomputers









ECP's work encompasses applications, system software, hardware technologies and architectures, and workforce development

The DOE ECP Plan of Record

- A **7-year project** that follows the *holistic/co-design* approach, which runs through 2023 (including 12 months schedule contingency)
  - To meet the ECP goals
- Enable an **initial exascale system** based on ***advanced architecture*** and delivered in **2021**
- Enable ***capable exascale*** **systems**, based on ECP R&D, delivered in **2022 and deployed in 2023** as part of an DOE facility upgrade
- Acquisition of the exascale systems is outside of the ECP scope, will be carried out by DOE facilities

Funding for ECP Application, Co-design Center, and Software Project


NEWS RELEASE

EXASCALE COMPUTING PROJECT

The Exascale Computing Project Awards $34 Million for Software Development

OAK RIDGE, Tenn., Nov. 10, 2016 – The Department of Energy's Exascale Com
(ECP) today announced the selection of 35 software development proposals
research and academic organizations.

The awards for the first year of funding total $34 million and cover many co
software stack for exascale systems, including programming models and run
mathematical libraries and frameworks, tools, lower-level system software,
and I/O, as well as in situ visualization and data analysis.

NEWS RELEASE

EXASCALE COMPUTING PROJECT

For Immediate Distribution

The Exascale Computing Project (ECP) Announces $39.8 million in First-Round Application Development Award

OAK RIDGE, Tenn., Sept. 07, 2016 – The Department of Energy's Exascale Computi
(ECP) today announced its first round of funding with the selection of 15 applicatio
development proposals for full funding and seven proposals for seed funding, repr
teams from 45 research and academic organizations.

The awards, totaling $39.8 million, target advanced modeling and simulation solut
specific challenges supporting key DOE missions in science, clean energy and natio
as well as collaborations such as the Precision Medicine Initiative with the Nation
Health's National Cancer Institute.

EWS RELEASE

EXASCALE COMPUTING PROJECT

The Exascale Computing Project Announces $48 Million to Establish Four Exascale Co-Design Centers

OAK RIDGE, Tenn., Nov. 11, 2016 – The Department of Energy's Exascale Computing Project (ECP) today
announced that it has selected four co-design centers as part of a 4 year, $48 million funding award. The first
year is funded at $12 million, and is to be allocated evenly among the four award recipients.

he ECP is responsible for the planning, execution, and delivery of technologies necessary for a capable
xascale ecosystem to support the nation's exascale imperative including software, ...
nd early testbed platforms.

EXASCALE COMPUTING PROJECT

ECP at UTK/ICL Involved in 7 Projects

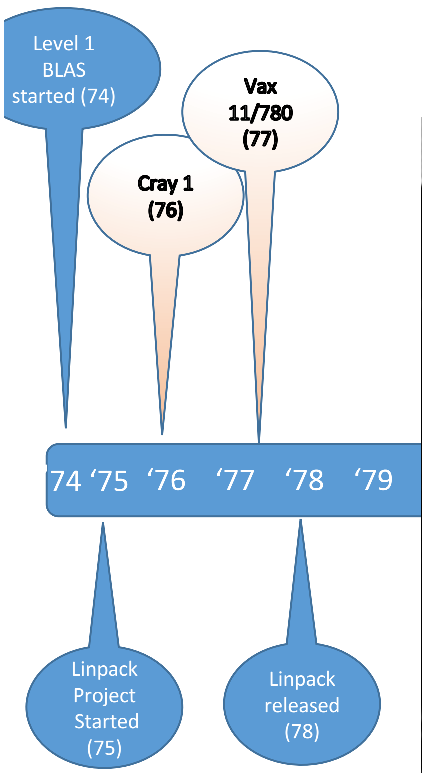
- Software Technology (35 funded projects); UTK/ICL is participating in…
  - **SLATE – provides SOA algorithmic and technology innovation in dense linear algebra software**
  - **EXA-PAPI - provides tool designers and application engineers with a consistent interface and methodology for the use of low-level performance counter hardware found across the system**
  - **PaRSEC - provides a runtime component to dynamically execute on heterogeneous distributed systems**
  - OMPI – provides MPI for exascale through improvements in scalability, capability, and resilience.
  - XSDK – provides interoperability across existing numerical libraries hypre, PETSc, SuperLU, Trilinos, MAGMA, PLASMA and DPLASMA
  - Peeks – provides interfaces and fundamental sparse kernels to make future GPU solvers and latency-tolerant solvers possible as configure-time plugins into Trilinos.
- Co-Design Centers (4 funded projects): UTK/ICL participating in …
  - CEED - Co-Design next-generation discretization software and algorithms that will enable a wide range of FE applications

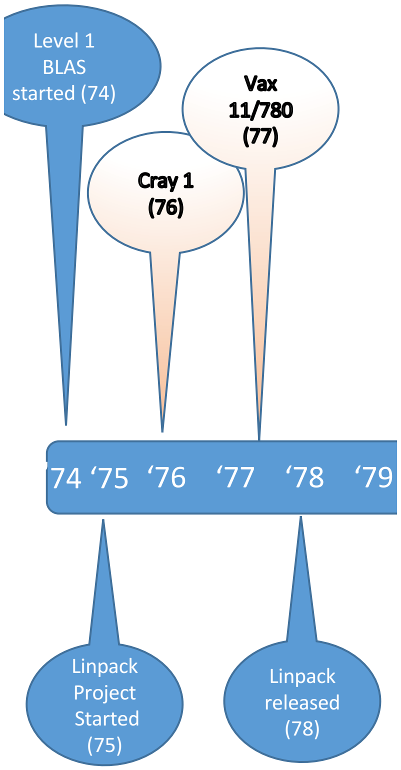
Software for Linear Algebra Targeting Exascale (SLATE)
Focused on Dense Linear Algebra Problems

- Linear systems of equations $Ax = b$
- Linear least squares $\min \| b - Ax \|_2$
- Singular value decomposition (SVD) $A = U\Sigma V^T$
- Eigenvalue value problems (EVP) $Ax = \lambda x$

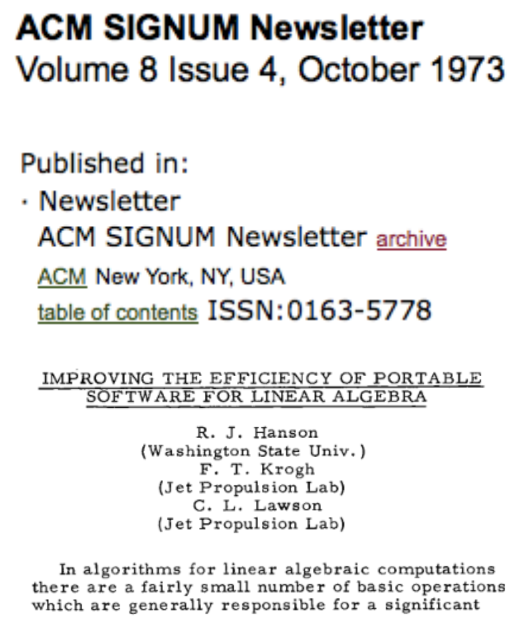
- Dense (square, rectangular)
- Band

But first, let's go back in time.


Level 1 BLAS started (74)
Cray 1 (76)
Vax 11/780 (77)
'74 '75 '76 '77 '78 '79
Linpack Project Started (75)
Linpack released (78)



- 1974: Effort to standardize Basic Linear Algebra Subprograms
  - Basic LA vector operations
  - Referred to now as Level 1 BLAS
- 1975: LINPACK Project started
  - Effort to produce portable, efficient linear algebra software for dense matrix computations.
- 1976: Vector computers in use for HPC
- 1977: DEC VAX system in common use



ACM SIGNUM Newsletter
Volume 8 Issue 4, October 1973

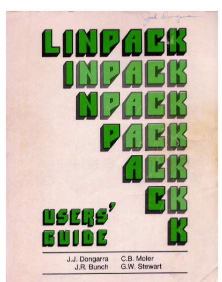
Published in:
- Newsletter

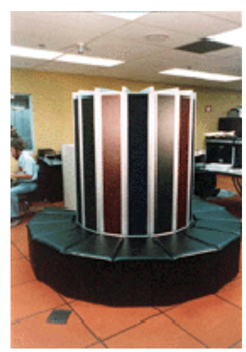
ACM SIGNUM Newsletter archive
ACM New York, NY, USA
table of contents ISSN:0163-5778

IMPROVING THE EFFICIENCY OF PORTABLE SOFTWARE FOR LINEAR ALGEBRA

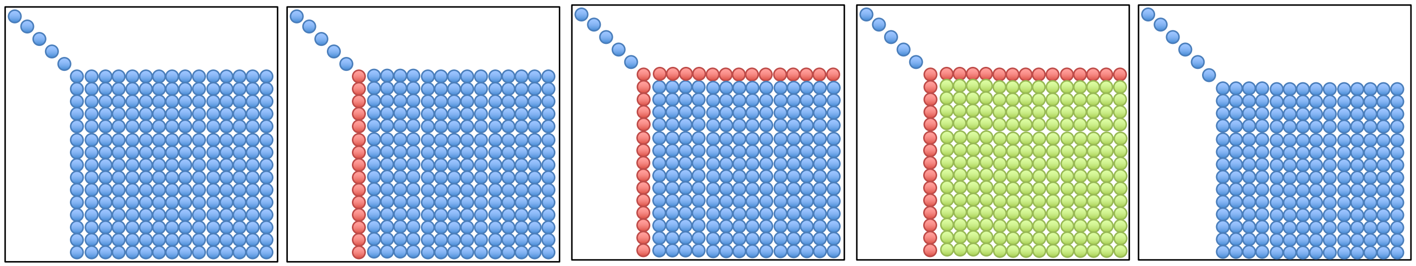
R. J. Hanson
(Washington State Univ.)
F. T. Krogh
(Jet Propulsion Lab)
C. L. Lawson
(Jet Propulsion Lab)

In algorithms for linear algebraic computations there are a fairly small number of basic operations which are generally responsible for a significant





The Standard LU Factorization LINPACK
1970's HPC of the Day: Vector Architecture



Factor column with Level 1 BLAS

Divide by Pivot row

Schur complement update (Rank 1 update)

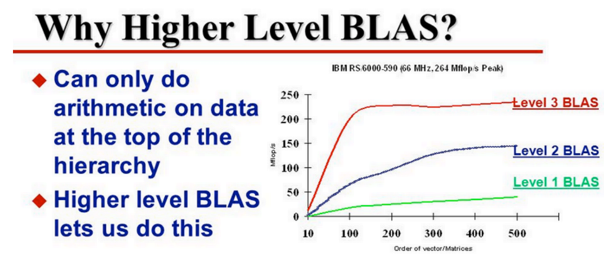
Next Step

Main points

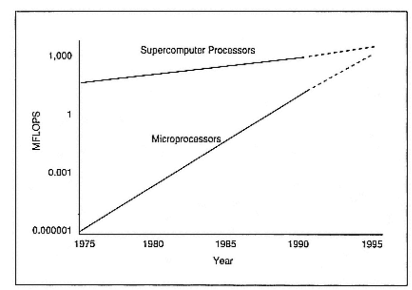
- Factorization column (zero) mostly sequential due to memory bottleneck
- Level 1 BLAS
- Divide pivot row has little parallelism
- OK on machines with excess memory bandwidth, but
- Too much data movement per step

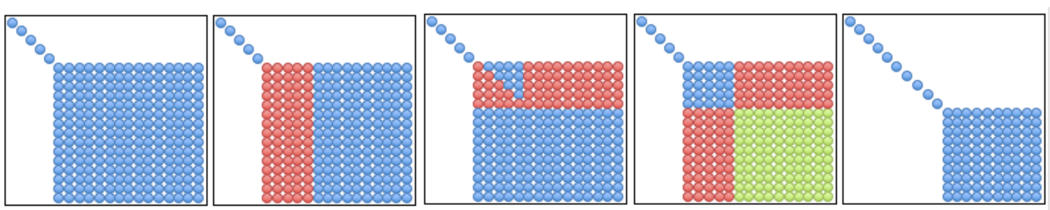
1984 - 1990

- Level 3 BLAS standardization started
- Level 2 BLAS standard published
- “Attack of the Killer Micros”, Brooks @ SC90
- Cache based & SMP machines
- Blocked partitioned algorithms was the way to go
  - Reduce data movement; today’s buzzword “Communication avoiding”



BLAS	Memory Refs	Flops	Flops/ Memory Refs
Level 1 $y=y+\alpha x$	$3n$	$2n$	$2/3$
Level 2 $y=y+Ax$	n^2	$2n^2$	2
Level 3 $C=C+AB$	$4n^2$	$2n^3$	$n/2$



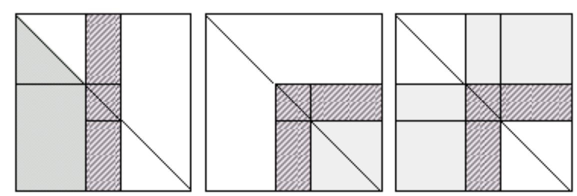


Factor panel (Level 1,2 BLAS)

Triangular Update (Level 3 BLAS)

Schur complement update (Level 3 BLAS)

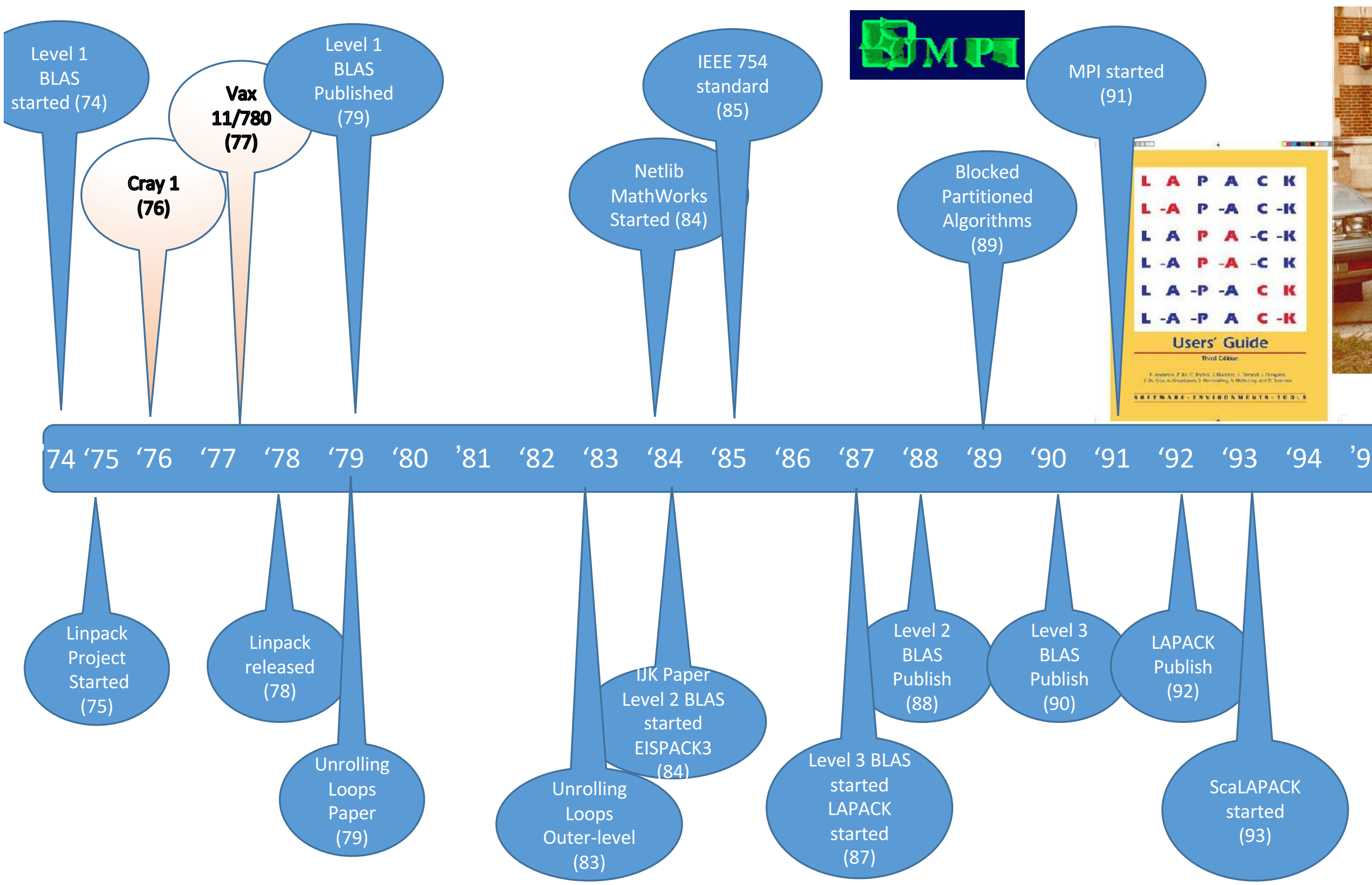
Next Step



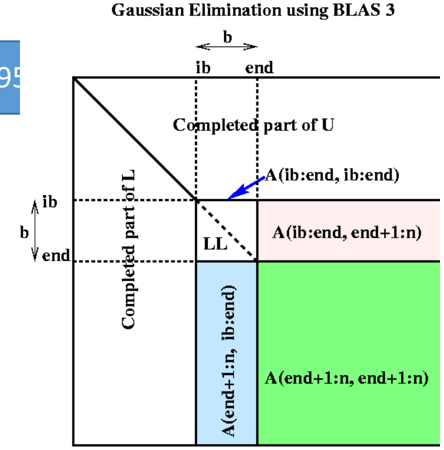
Left-looking LU

Right-looking LU

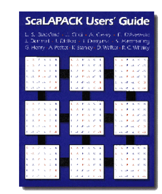
Crout LU



Gaussian Elimination using BLAS 3



- LAPACK Published
- ScaLAPACK started



LAPACK Functionality

Type of Problem	Acronyms
Linear system of equations	SV
Linear least squares problems	LLS
Linear equality-constrained least squares problem	LSE
General linear model problem	GLM
Symmetric eigenproblems	SEP
Nonsymmetric eigenproblems	NEP
Singular value decomposition	SVD
Generalized symmetric definite eigenproblems	GSEP
Generalized nonsymmetric eigenproblems	GNEP
Generalized (or quotient) singular value decomposition	GSVD (QSVD)

LAPACK Software

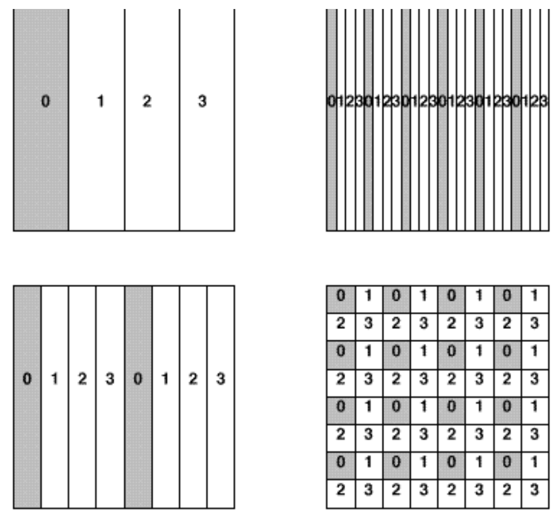
Jointly with UTK and UCB and Many Other Contributors

- First release in February 1992 (Silver Anniversary)
- Current: LAPACK Version 3.7.0 (Dec, 2017) ~2M LoC
- **LICENSE:** Mod-BSD, freely-available software package - Thus, it can be included in commercial software packages (and has been). We only ask that proper credit be given to the authors.
- Public GITHub repository
- **4 Precisions:** single, double, complex, double complex
  - *Considering 16-bit flpt version*
- **Multi-OS** *nix, Mac OS/X, Windows
- **Multi-build** support (Make and Cmake)
- **Reference BLAS and CBLAS**
- **LAPACKE: Standard C language APIs for LAPACK**
- Prebuilt Libraries for **Windows**
- Extensive test suite
- **Forum and User support:** http://icl.cs.utk.edu/lapack-forum/
- Goal: bug free library – Since 2009, 165 bugs reported, only 11 pending correction

ScaLAPACK

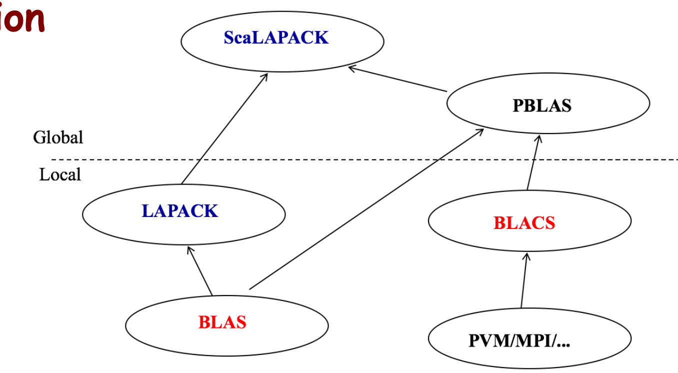
- Library of software dealing with dense & banded routines
- Distributed Memory - Message Passing
- MIMD Computers and Networks of Workstations
- Clusters of SMPs
- Data layout critical for performance

- Relies on LAPACK / BLAS and BLACS / MPI
- Includes PBLAS (Parallel BLAS)



Programming Style

- SPMD Fortran 77 using an object based design
- Built on various modules
  - PBLAS Interprocessor communication & computation
    - BLAS
    - BLACS
      - MPI, PVM, IBM SP, CRI T3, Intel, TMC
      - Provides right level of abstraction.



- Object based - Array descriptor
  - Contains information required to establish mapping between a global array entry and its corresponding process and memory location.
  - Provides a flexible framework to easily specify additional data distributions or matrix types.
  - Currently dense, banded, & out-of-core
- Using the concept of context

PBLAS

- Parallel Basic Linear Algebra Subprograms for distributed-memory MIMD computers
- Do both the communication and computation, but done in phases.
- Simplification of the parallelization: especially when BLAS-based,
- Modularity: gives programmer larger building blocks,
- Portability: machine dependencies are confined to the BLAS and BLACS the computation and communication phases.
- Global view of the matrix operands, allowing global addressing of distributed matrices (hiding complex local indexing),
- Fits with the distribution patterns for High Performance Fortran (HPF)
- Load balance maintained

BLACS – Basic Linear Algebra Communication Subprograms

- A design tool, **they are a conceptual aid in design and coding.**
- **Associate widely recognized mnemonic names with communication operations, improve**
  - program readability,
  - self-documenting quality **of the code.**
- **Promote** efficiency **by identifying frequently occurring operations of linear algebra which can be optimized on various computers.**

It allows the user to

- **create arbitrary groups of processes,**
- **create multiple overlapping and/or disjoint grids,**
- **isolate each process grid so that grids do not interfere with each other.**

BLACS context ⟷ MPI communicator

PBLAS

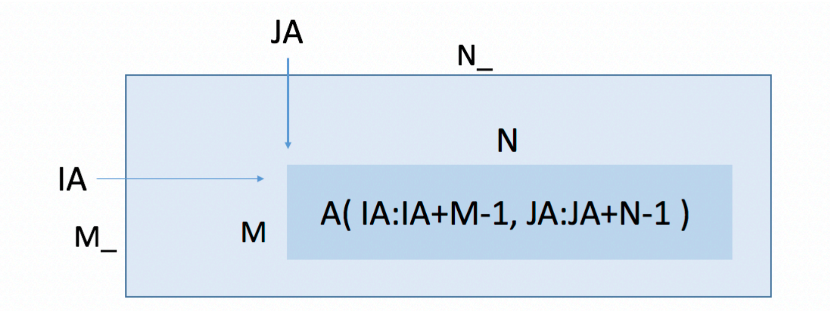
- Similar to the BLAS in functionality and naming.
- Built on the BLAS and BLACS
- Provide global view of matrix

```
CALL DGEXXX  ( M, N, A( IA, JA ), LDA,... )
```

⇩

```
CALL PDGEXXX( M, N, A, IA, JA, DESCA,... )
```

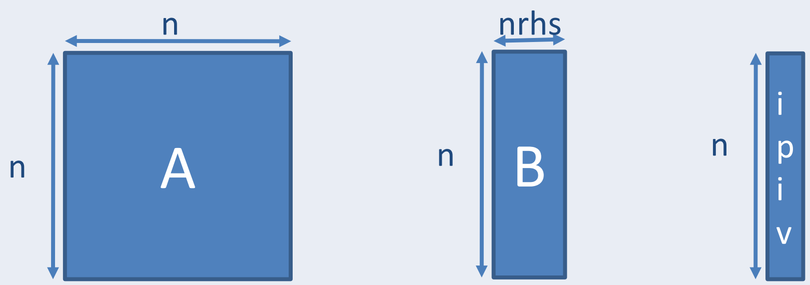
- Hides complex local indexing



From LAPACK to ScaLAPACK

[LAPACK] subroutine dgesv(n, nrhs, a(ia,ja), lda, ipiv, b(ib,jb), ldb, info)

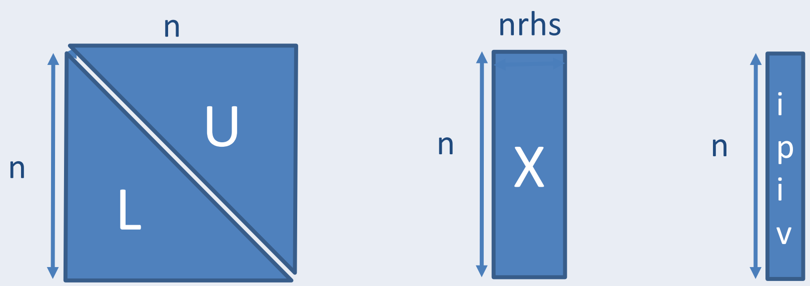
input:



LAPACK Data layout

info

output:



LAPACK Data layout

info

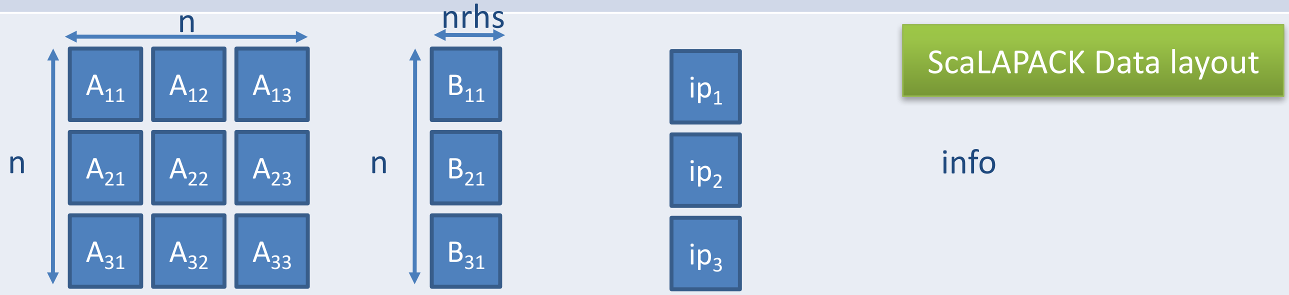
From LAPACK to ScaLAPACK

[LAPACK] subroutine dgesv(n, nrhs, a(ia,ja), lda, ipiv, b(ib,jb), ldb, info)

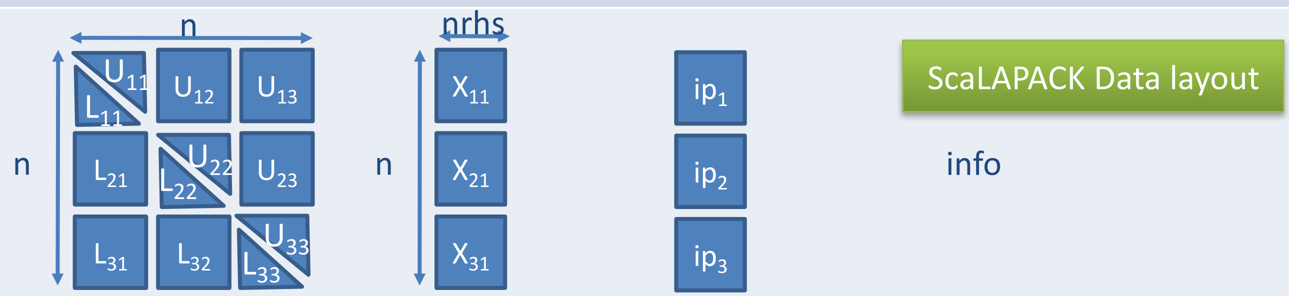
From LAPACK to ScaLAPACK

[LAPACK] subroutine dgesv(n, nrhs, a(ia,ja), lda, ipiv, b(ib,jb), ldb, info)
[ScaLAPACK] subroutine pdgesv(n, nrhs, a, ia, ja, desca, ipiv, b, ib, jb, descb, info)

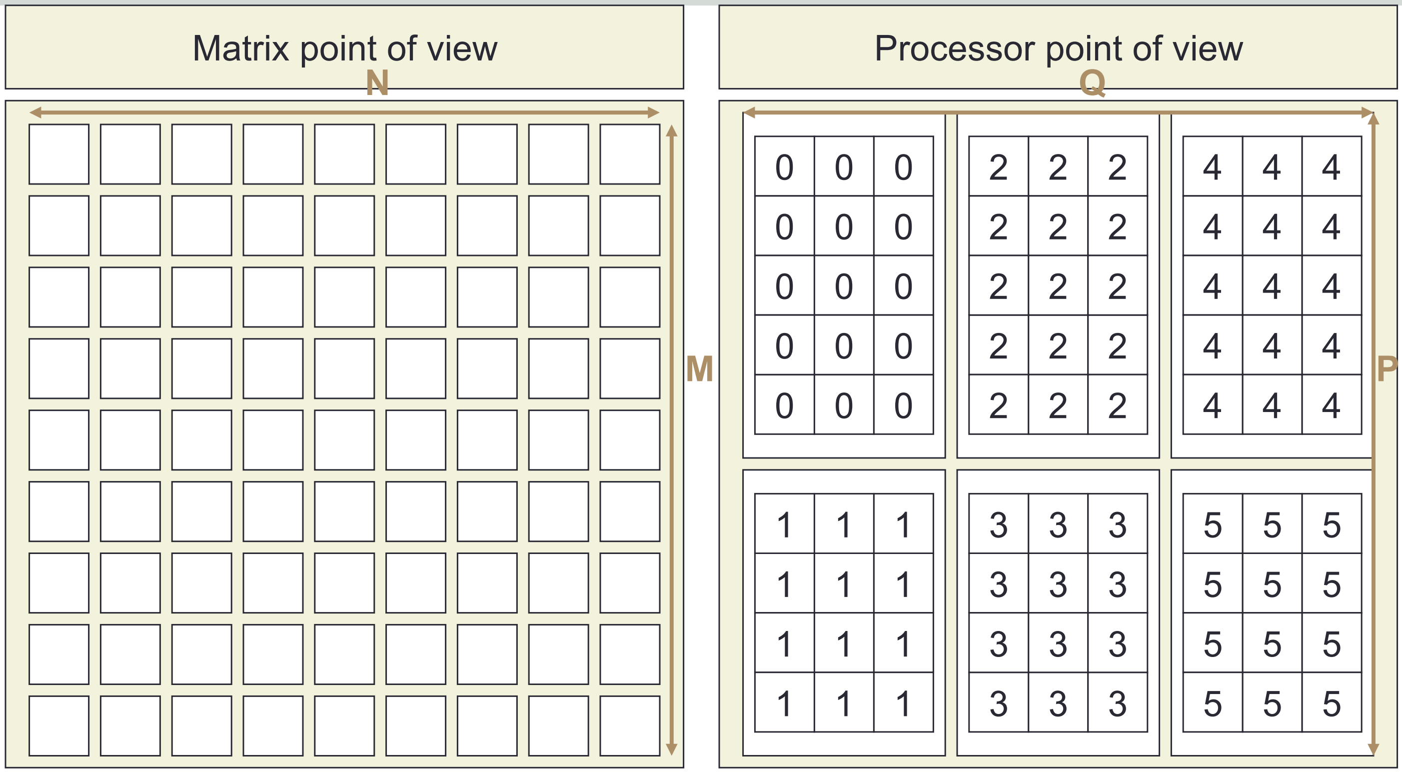
input:

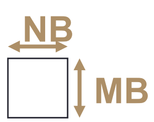


output:



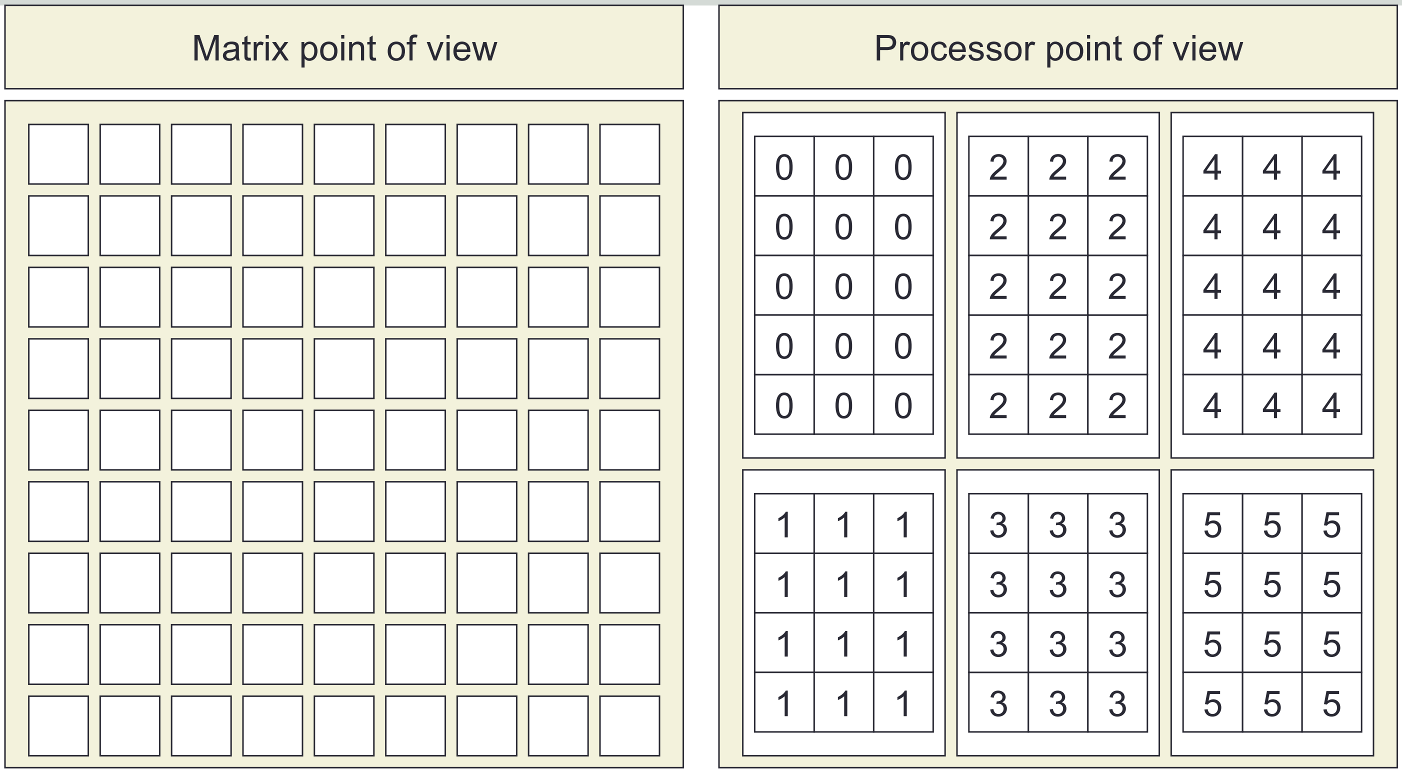
2D Block Cyclic Layout

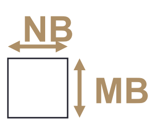




Matrix is MxN
Process grid is PxQ, P=2, Q=3
Blocks are MBxNB

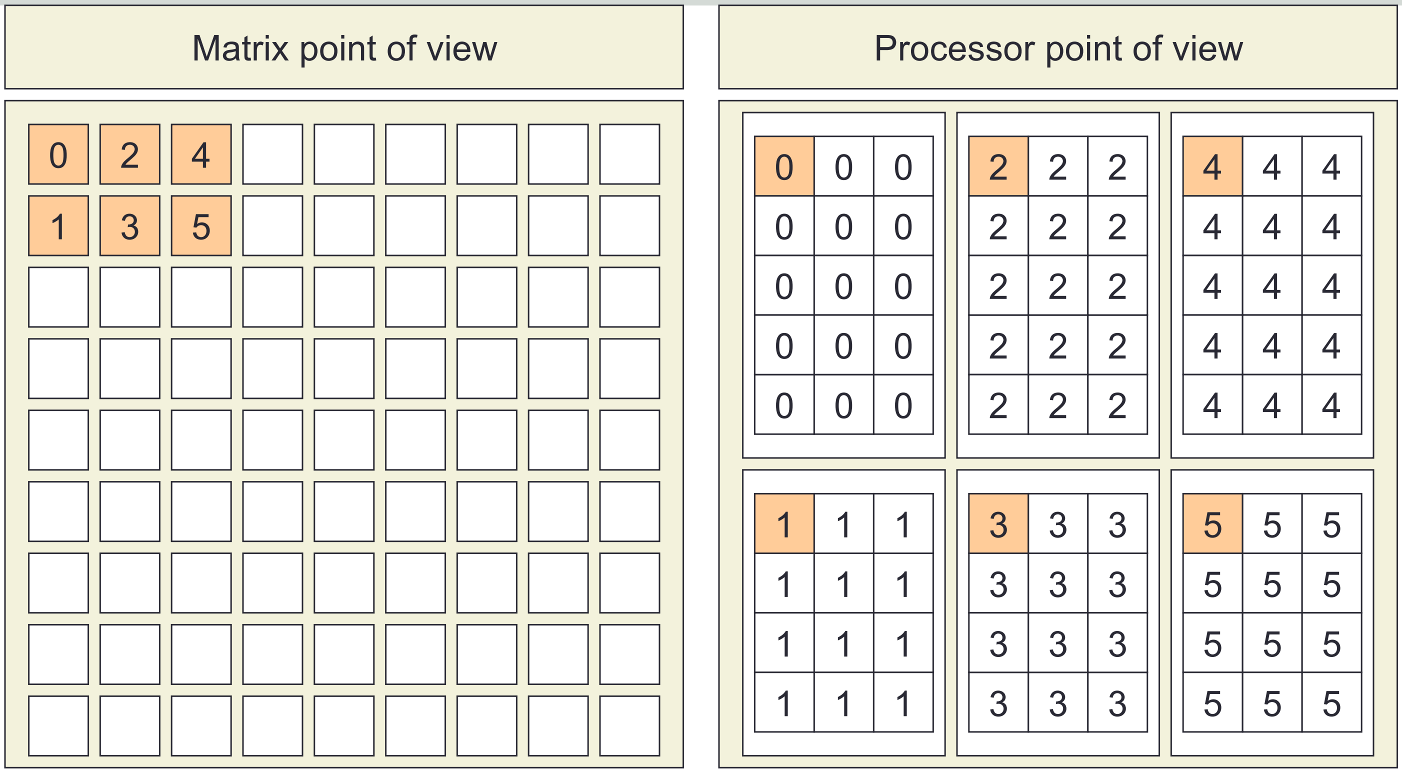
2D Block Cyclic Layout

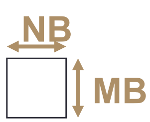




Matrix is MxN
Process grid is PxQ, P=2, Q=3
Blocks are MBxNB

2D Block Cyclic Layout





Matrix is MxN
Process grid is PxQ, P=2, Q=3
Blocks are MBxNB

2D Block Cyclic Layout

Matrix point of view

0	2	4	0	2	4			
1	3	5	1	3	5			

Processor point of view

0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

2	2	2
2	2	2
2	2	2
2	2	2
2	2	2

4	4	4
4	4	4
4	4	4
4	4	4
4	4	4

1	1	1
1	1	1
1	1	1
1	1	1

3	3	3
3	3	3
3	3	3
3	3	3

5	5	5
5	5	5
5	5	5
5	5	5

2D Block Cyclic Layout

Matrix point of view

0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4

Processor point of view

0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

2	2	2
2	2	2
2	2	2
2	2	2
2	2	2

4	4	4
4	4	4
4	4	4
4	4	4
4	4	4

1	1	1
1	1	1
1	1	1
1	1	1

3	3	3
3	3	3
3	3	3
3	3	3

5	5	5
5	5	5
5	5	5
5	5	5

2D Block Cyclic Layout

Matrix point of view

0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4

Processor point of view

0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4

1	1	1	3	3	3	5	5	5
1	1	1	3	3	3	5	5	5
1	1	1	3	3	3	5	5	5
1	1	1	3	3	3	5	5	5

2D Block Cyclic Layout

Matrix point of view

0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4

Processor point of view

0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

2	2	2
2	2	2
2	2	2
2	2	2
2	2	2

4	4	4
4	4	4
4	4	4
4	4	4
4	4	4

1	1	1
1	1	1
1	1	1
1	1	1

3	3	3
3	3	3
3	3	3
3	3	3

5	5	5
5	5	5
5	5	5
5	5	5

2D Block Cyclic Layout

Matrix point of view

3	5	1	3	5	1	3	5
2	4	0	2	4	0	2	4
3	5	1	3	5	1	3	5
2	4	0	2	4	0	2	4
3	5	1	3	5	1	3	5
2	4	0	2	4	0	2	4
3	5	1	3	5	1	3	5
2	4	0	2	4	0	2	4

Processor point of view

0	0
0	0
0	0
0	0

2	2	2
2	2	2
2	2	2
2	2	2

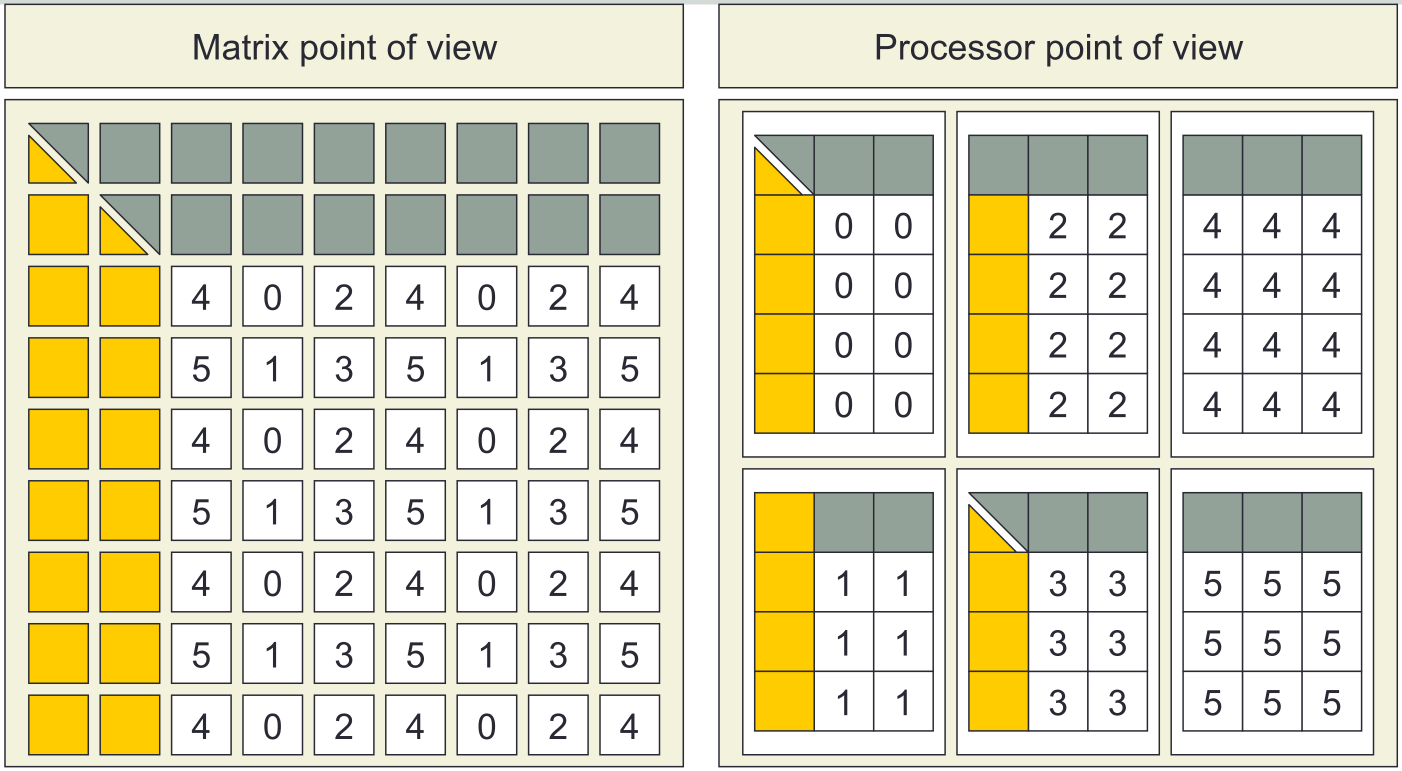
4	4	4
4	4	4
4	4	4
4	4	4

1	1
1	1
1	1
1	1

3	3	3
3	3	3
3	3	3
3	3	3

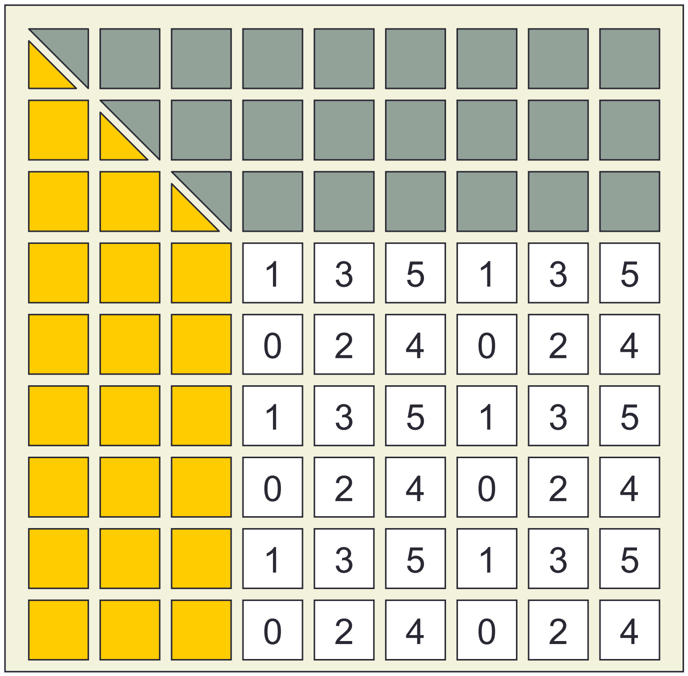
5	5	5
5	5	5
5	5	5
5	5	5

2D Block Cyclic Layout

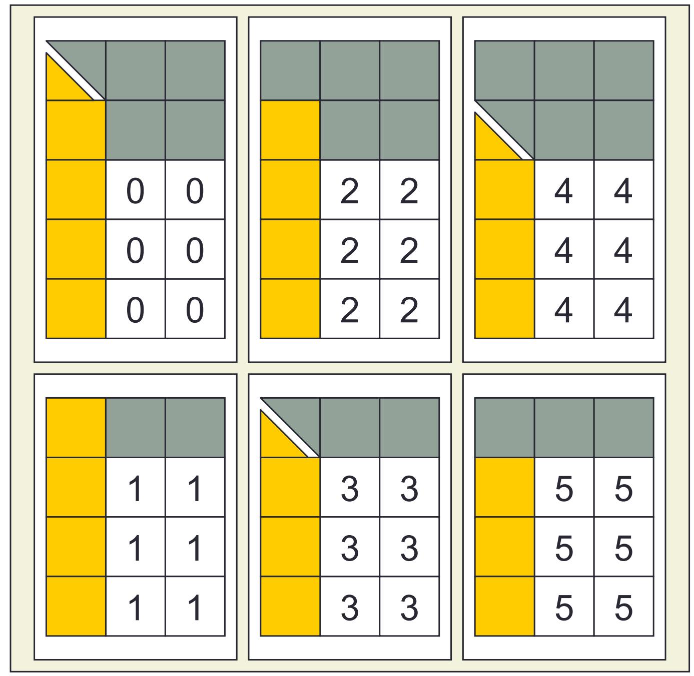


2D Block Cyclic Layout

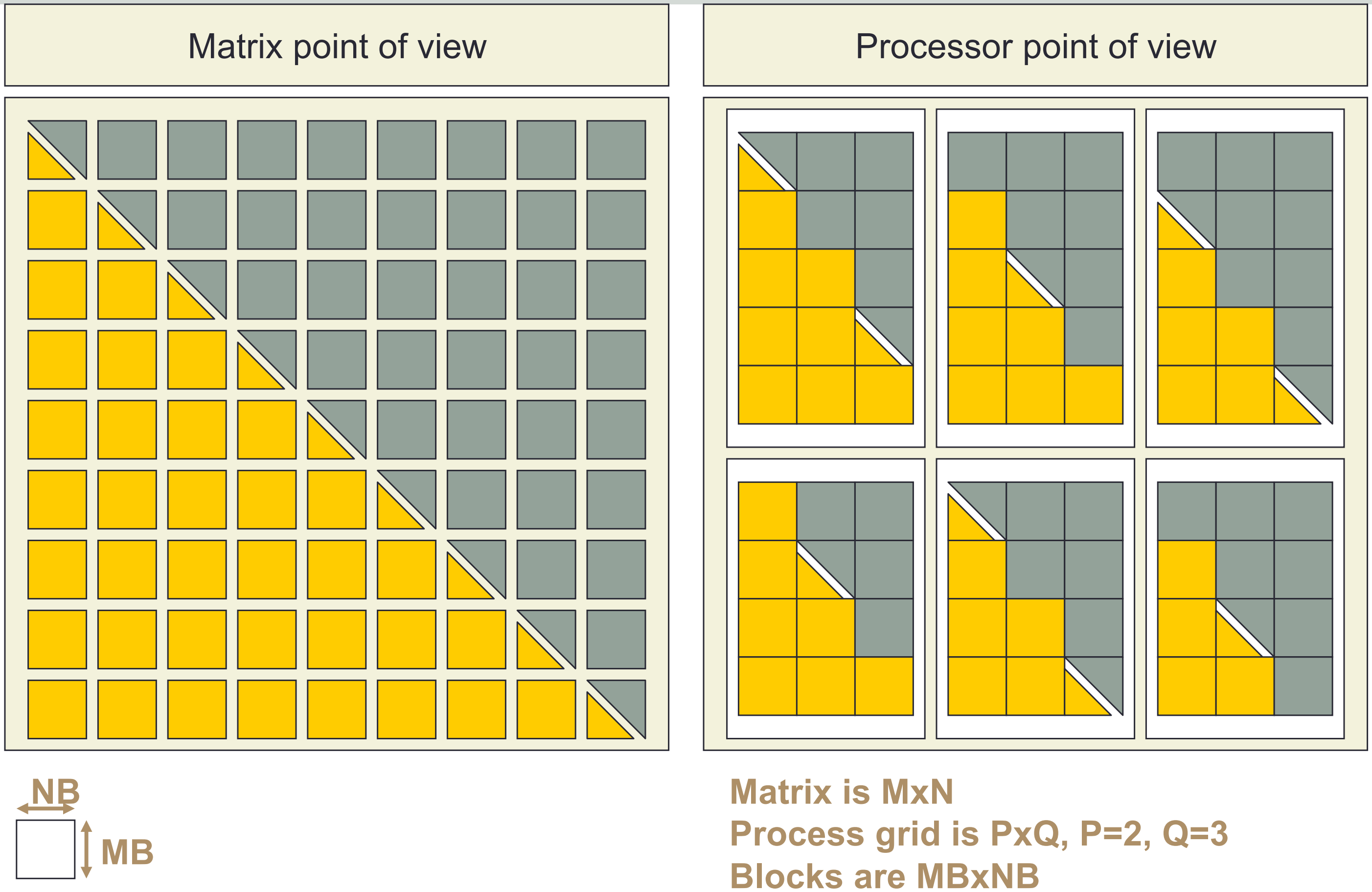
Matrix point of view



Processor point of view



2D Block Cyclic Layout



Matrix is MxN
Process grid is PxQ, P=2, Q=3
Blocks are MBxNB

LAPACK Functionality

Type of Problem	Acronyms
Linear system of equations	SV
Linear least squares problems	LLS
Linear equality-constrained least squares problem	LSE
General linear model problem	GLM
Symmetric eigenproblems	SEP
Nonsymmetric eigenproblems	NEP
Singular value decomposition	SVD
Generalized symmetric definite eigenproblems	GSEP
Generalized nonsymmetric eigenproblems	GNEP
Generalized (or quotient) singular value decomposition	GSVD (QSVD)

ScaLAPACK Functionality

Type of Problem	Acronyms
Linear system of equations	SV
Linear least squares problems	LLS
~~Linear equality-constrained least squares problem~~	~~LSE~~
~~General linear model problem~~	~~GLM~~
Symmetric eigenproblems	SEP
Nonsymmetric eigenproblems	NEP
Singular value decomposition	SVD
~~Generalized symmetric definite eigenproblems~~	~~GSEP~~
~~Generalized nonsymmetric eigenproblems~~	~~GNEP~~
~~Generalized (or quotient) singular value decomposition~~	~~GSVD (QSVD)~~

Performance Issues with ScaLAPACK

- The major problem with ScaLAPACK is the lack of overlap of computation and communication .
- Each phase done separately, bulk synchronous.
  - Computation phase then a communication phase.
  - All (most) processes compute then a communication phase (broadcast)
  - This is how the PBLAS operate.
- No overlap, resulting in performance issues

- Need an “new” interface which allows computation and communication to take place simultaneously, in an asynchronous fashion.

Problems with Sca/LAPACK

software engineering

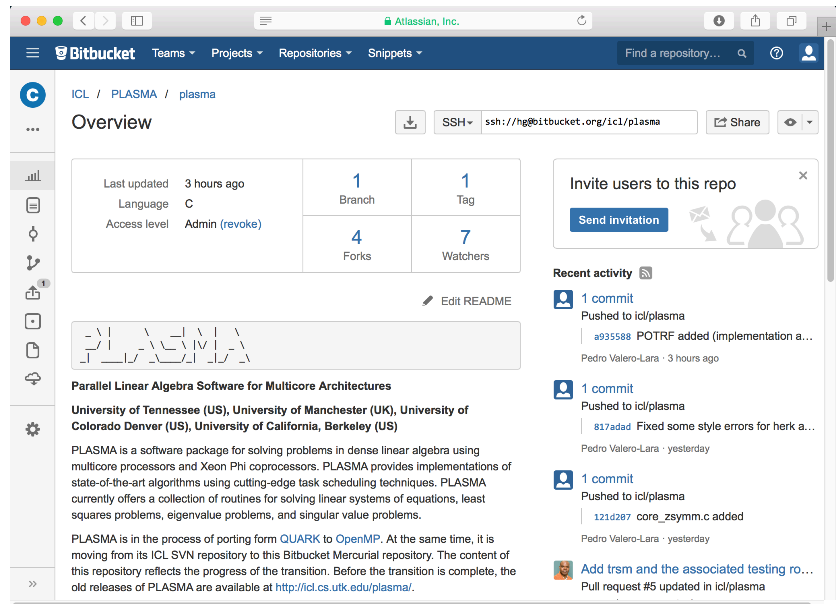
- Obsolete language (F77)
  - Poor man's object orientation (array descriptors)
  - Manual generation of 4 precisions
    - Hard to accommodate lower (e.g. half) or higher (e.g. double-double)
  - No convenience of memory allocation (e.g. workspaces)
  - Hard to maintain with C/C++ educated personnel

Since LAPACK and ScaLAPACK

- A lot has changed
  - Manycore and accelerators
  - Use a different set of ideas to provide efficient use of underlying hardware
    - PLASMA/DPLASMA
    - MAGMA

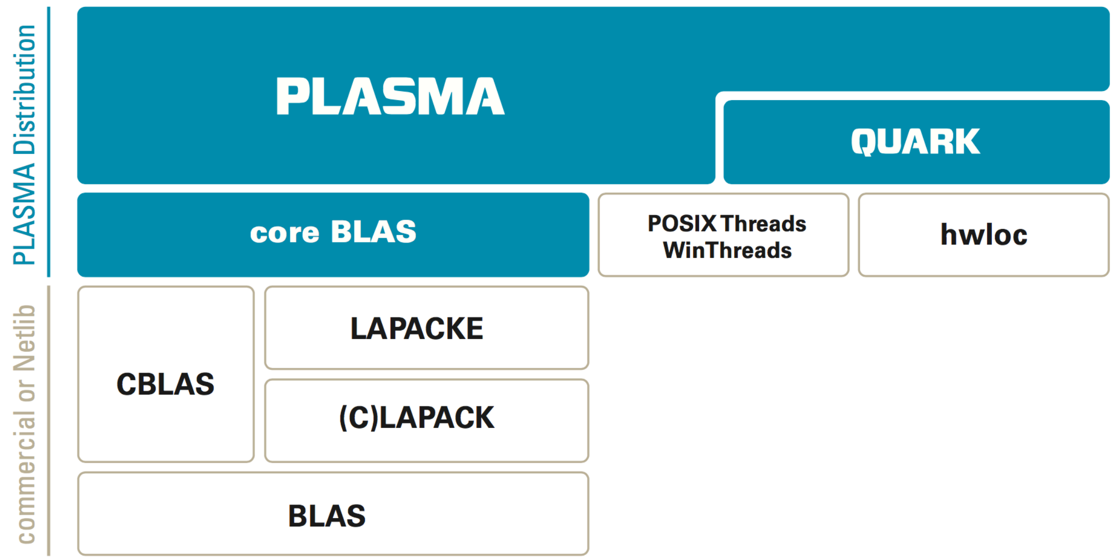
PLASMA

- Dense linear algebra
  - linear systems of equations
  - linear least squares
  - singular value decomposition
  - eigenvalue problems (symmetric)
- Ideally a replacement for LAPACK
- Multicore CPUs
- Xeon Phi

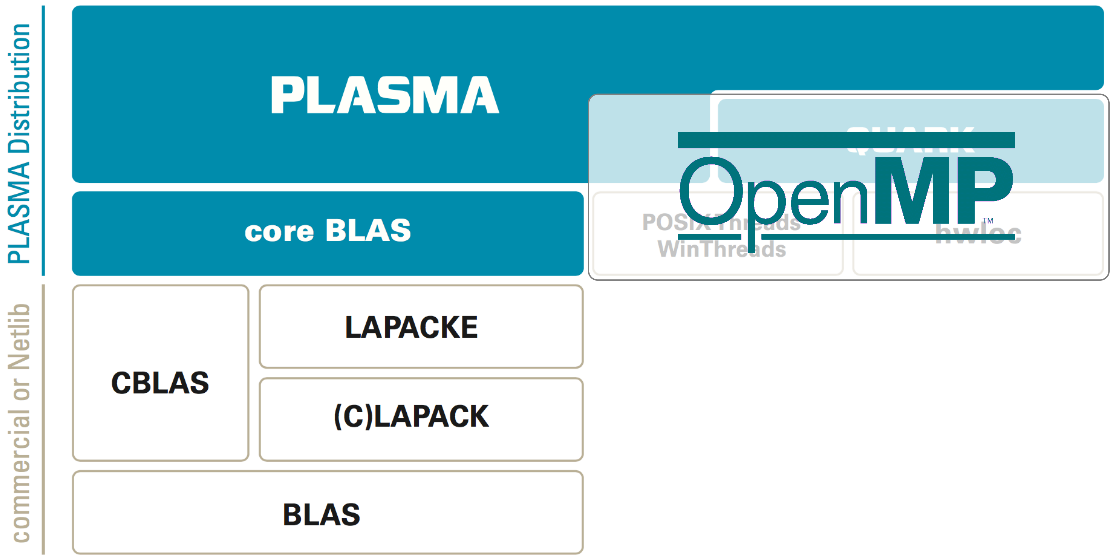


http://icl.cs.utk.edu/plasma/ → https://bitbucket.org/icl/plasma

PLASMA – Original Software Stack when the Project Started



PLASMA – Software Stack

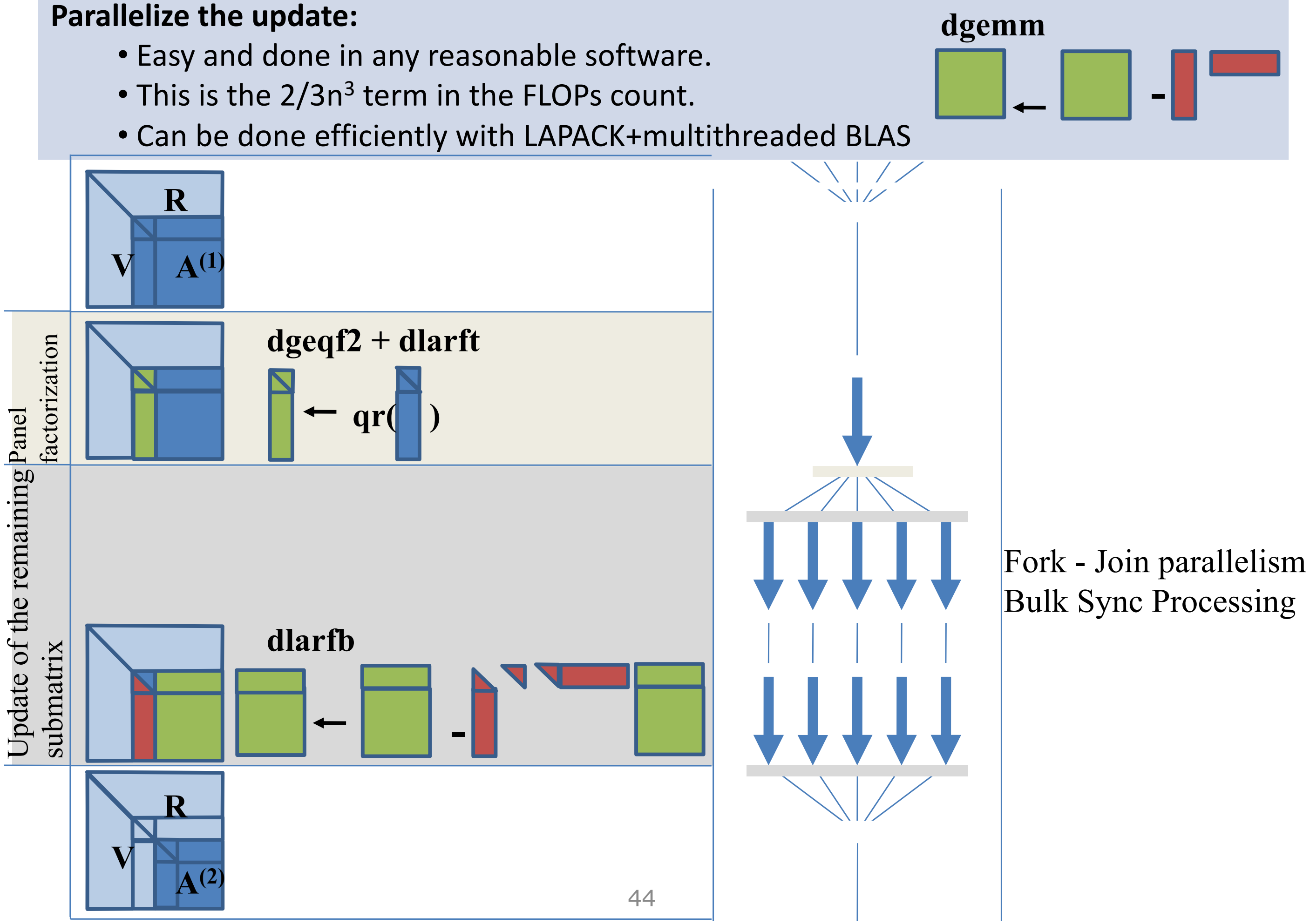


A. YarKhan, J. Kurzak, P. Luszczek, J. Dongarra, *Porting the PLASMA Numerical Library to the OpenMP Standard*, International Journal of Parallel Programming, 1-22, 2016.
[DOI: 10.1007/s10766-016-0441-6](https://doi.org/10.1007/s10766-016-0441-6)

Parallelization of QR Factorization

Parallelize the update:

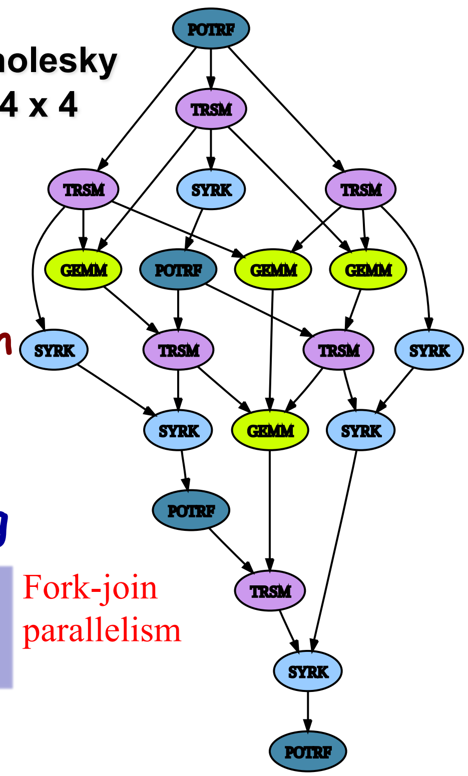
- Easy and done in any reasonable software.
- This is the $2/3n^3$ term in the FLOPs count.
- Can be done efficiently with LAPACK+multithreaded BLAS

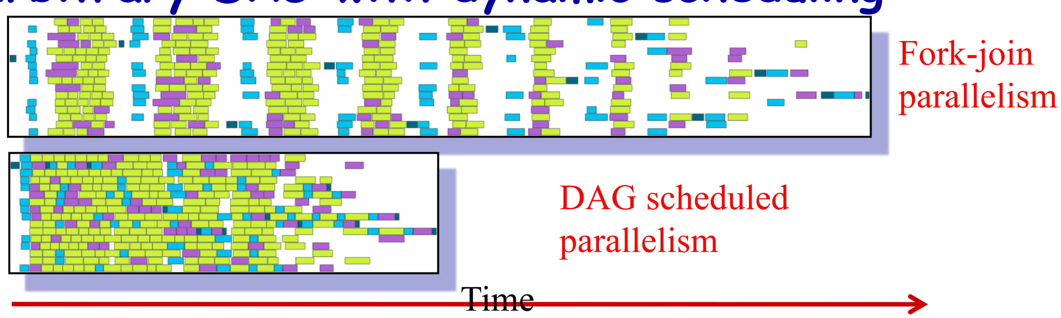


PLASMA: Parallel Linear Algebra s/w for Multicore Architectures

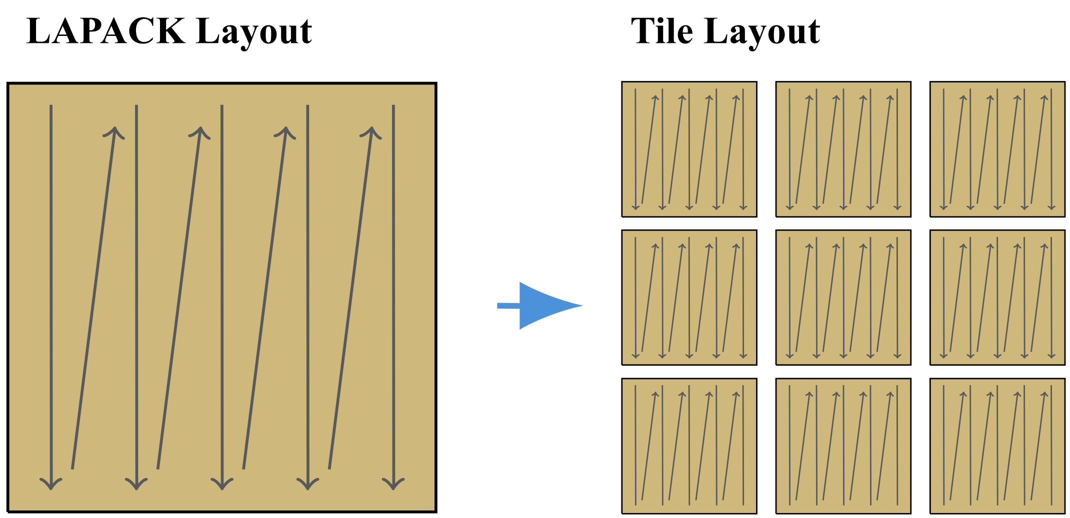
- **Objectives**
  - High utilization of each core
  - Scaling to large number of cores
  - Shared or distributed memory
- **Methodology**
  - Dynamic DAG scheduling
  - Split phases task generation and execution
  - Explicit parallelism/Implicit communication
  - Fine granularity / block data layout
- **Arbitrary DAG with dynamic scheduling**

Cholesky
4 x 4





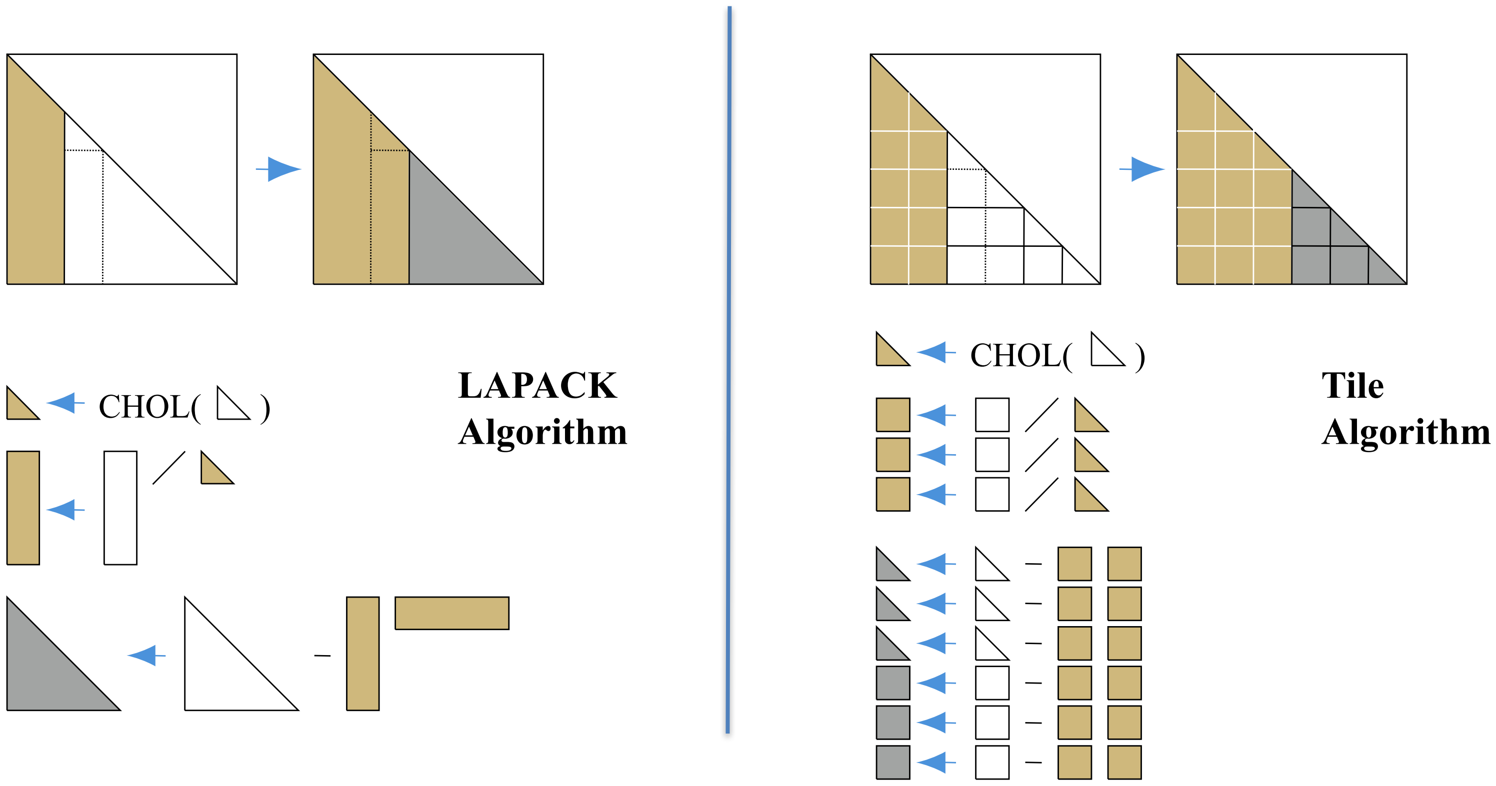
PLASMA – Tile Matrix Layout



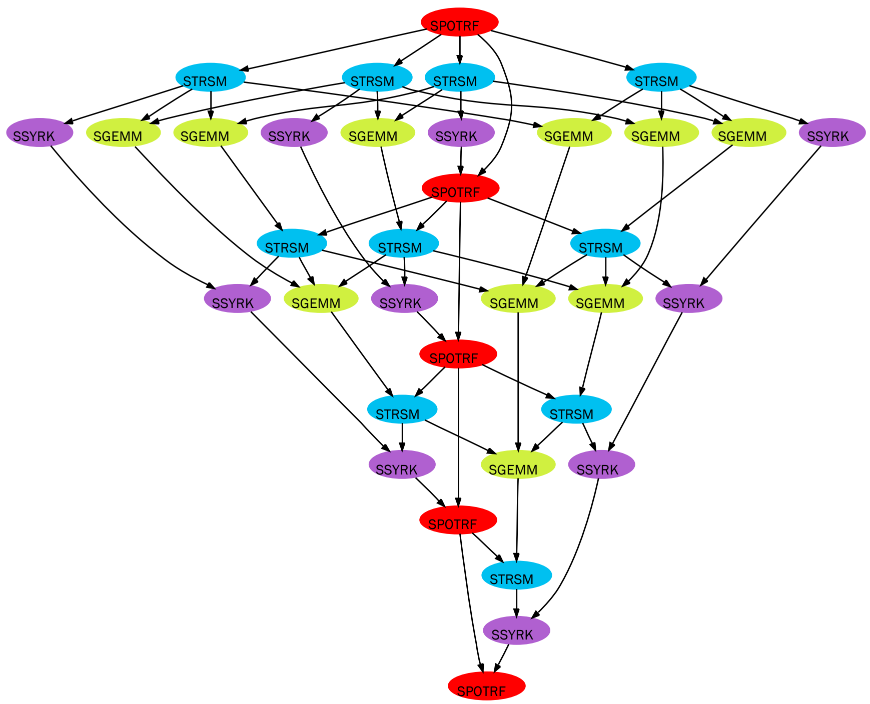
- enables dataflow scheduling
- helps memory efficiency
- simplifies communication

Translation can be done in place, in a parallel and cache efficient fashion.

PLASMA – Tile Algorithms



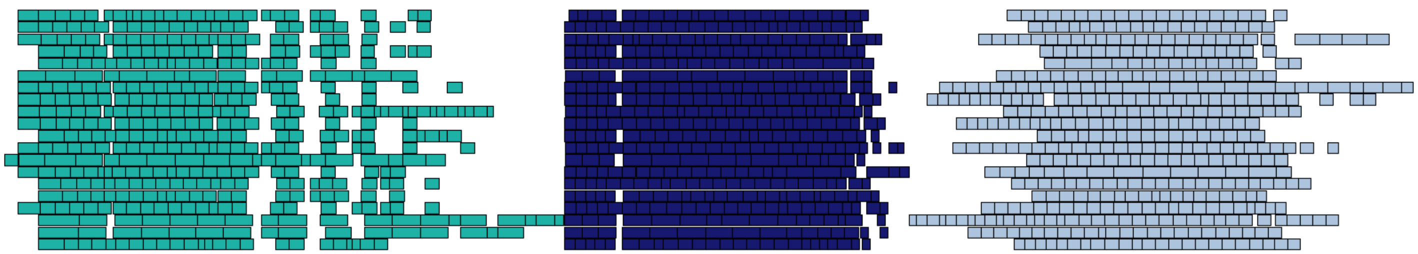
PLASMA – Dataflow Scheduling

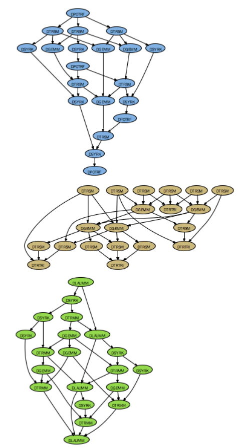


- Exploit parallelism
- Balance load
- Maximize data locality

PLASMA OpenMP – Inverse of the Variance-Covariance Matrix

PLASMA Cholesky inversion using OpenMP
Intel Xeon E5-2650 v3 (Haswell) 2.3GHz 20 cores
tiles of size 224 x 224, matrix of size 13 x 13 tiles (2912 x 2912)

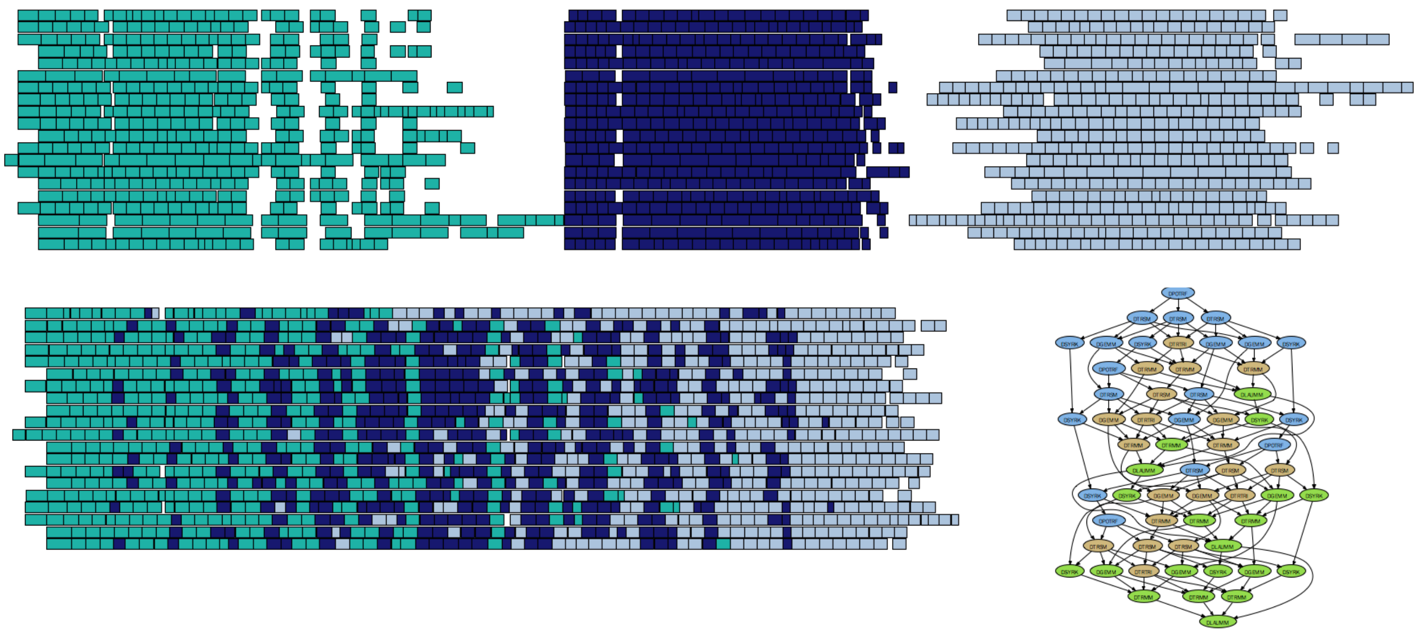


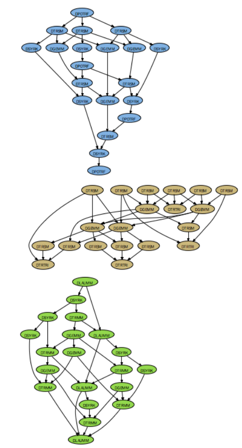


Factor matrix $A = LL^T$ | Compute inverse of factor L | Computer $A^{-1} = L^T L^{-1}$

PLASMA OpenMP – Inverse of the Variance-Covariance Matrix

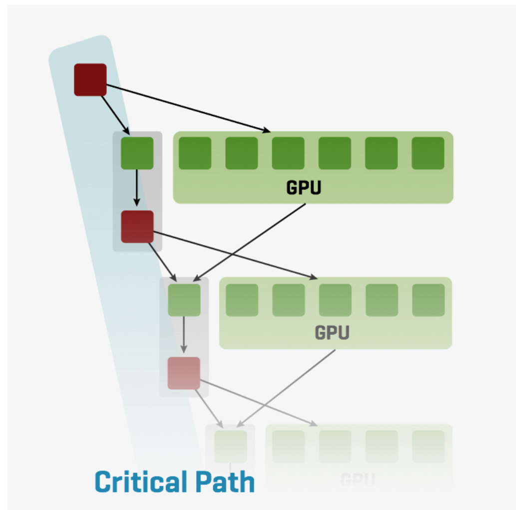
PLASMA Cholesky inversion using OpenMP
Intel Xeon E5-2650 v3 (Haswell) 2.3GHz 20 cores
tiles of size 224 x 224, matrix of size 13 x 13 tiles (2912 x 2912)





MAGMA – Focus on Using Accelerators

- Dense linear algebra for accelerators
  - NVIDIA using CUDA
  - AMD using OpenCL
  - Intel Xeon Phi
- Hybrid, CPU-GPU implementations
  - single-GPU
  - multi-GPU
  - OO-GPU-memory
- Managing data transfers
- Some batched routines
- Some sparse solvers



FEATURES AND SUPPORT

- MAGMA 2.2 FOR CUDA
- clMAGMA 1.4 FOR OpenCL
- MAGMA MIC 1.4 FOR Intel Xeon Phi

CUDA	OpenCL	Intel Xeon Phi	
●	●	●	Linear system solvers
●	●	●	Eigenvalue problem solvers
●	●		Auxiliary BLAS
●			Batched LA
●		●	Sparse LA
●	●	●	CPU Interface
●	●	●	GPU Interface
●	●	●	Multiple precision support
●			Non-GPU-resident factorizations
●	●	●	Multicore and multi-GPU support
●	●	●	LAPACK testing
●	●	●	Linux
●	●		Windows
●	●		Mac OS

MAGMA Routines Depending on where Matrix Located

Suffix	Example	Description
none	magma_dgesv	hybrid CPU/GPU routine – matrix in CPU memory
_m	**magma_dgesv_m**	hybrid CPU/multi-GPU routine – matrix in CPU memory
_gpu	**magma_dgesv_gpu**	hybrid CPU/GPU routine – matrix in GPU memory
_mgpu	**magma_dgesv_mgpu**	hybrid CPU/multi-GPU routine – matrix distributed across GPU memories

SLATE – Software for Linear Algebra Targeting Exascale

- Target Hardware DOE Exascale systems, as well as pre-Exascale
- Bring the best ideas of LAPACK, ScaLAPACK, PLASMA & MAGMA
- Goals
  - Efficiency – to run as fast as possible (close to theoretical peak);
  - Scalability – as the problem size and number of processors grow;
  - Reliability – including error bounds and rigorous LAPACK-derived testing suites;
  - Portability – across all important parallel machines (as described above);
  - Flexibility – so users can construct new routines from well-designed parts;
  - Ease of use – by making the interfaces look as similar as possible to LAPACK and ScaLAPACK.

SLATE: New Abstraction Layer

standardized components

Basically leverage the accomplishments of the last decade in modernization of the MPI and OpenMP standards, and the appearance of OpenCL and OpenACC, to create an abstraction layer suitable for supporting ScaLAPACK workloads and beyond.

- MPI 3
  - Non-blocking collectives
  - Neighborhood collectives
  - Improved one-sided communication
  - Thread friendliness
    - Thread-safe probe and receive
- OpenMP 4 / OpenACC
  - Accelerator offload
    - With memory management
  - Dynamic task scheduling
    - With data dependencies tracking
- OpenCL / OpenACC
  - Portable accelerator kernels

SLATE: New Abstraction Layer

programing frameworks

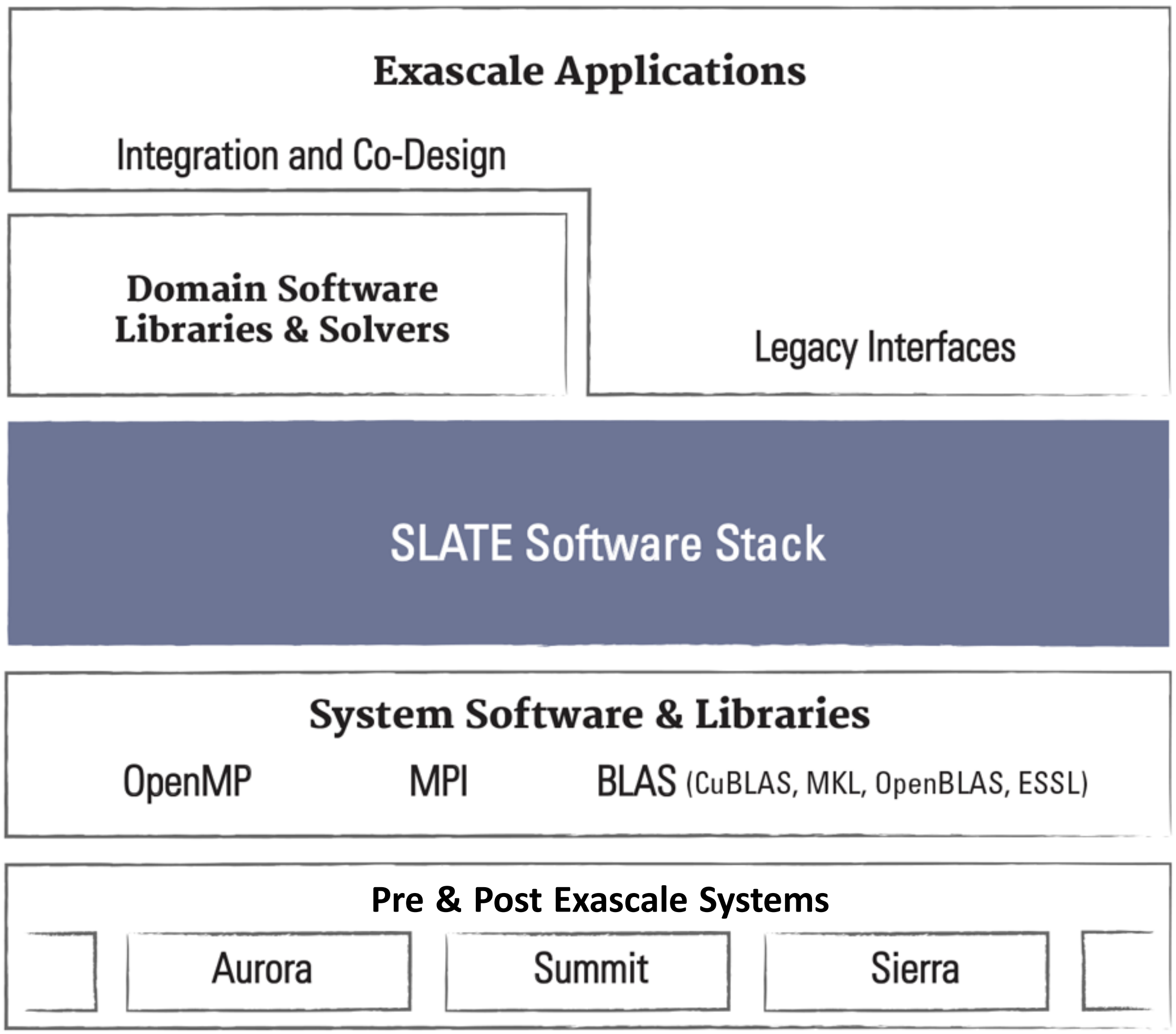
Leverage emerging programming frameworks for scheduling tasks to large scale machines with multicores, accelerators and complex memory systems.
Perhaps plug into different run-time systems

- Runtime will provide…
  - Dynamic task scheduling
    - Mutithreading
    - Accelerator offload
  - Accelerator memory management
    - Basically a cache model with LRU policy
  - Communication hiding
    - Asynchronous message passing
    - Asynchronous PCI DMAs (host-device)
  - Separation of concerns
    - Flexible task assignment
    - Flexible data assignment
- PaRSEC (UTK), StarPU (INRIA), Kokkos (SNL), Legion (Stanford),…

SLATE: Adopt C++ at the ScaLAPACK Level

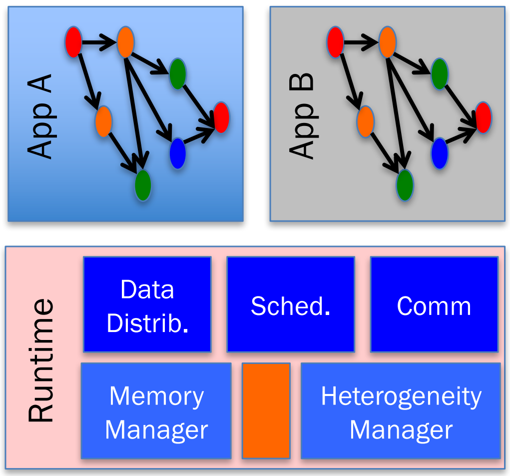
algorithmic level

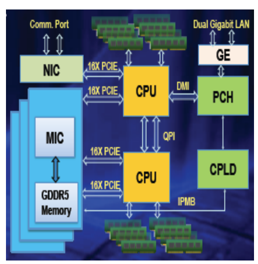
- Natural encapsulation
  - Intuitive objects
  - More coding safeties
- Natural polymorphism
  - Allow for multiple data layouts with no code duplication
- C++ templating
  - Easily deal with multiple precisions (Z, C, D, S)
  - Allow for adoption of half and extended
  - Cut the code base by 4x.
- Easily deal with dynamic memory allocation
- Exceptions
  - Much more compact error handling
- Provide classic (C, F77) interfaces if required



- **DOE ECP**
  - Two systems, one in 2021 and another in 2023
  - First: "advanced" architecture
  - Second: "capable" architecture

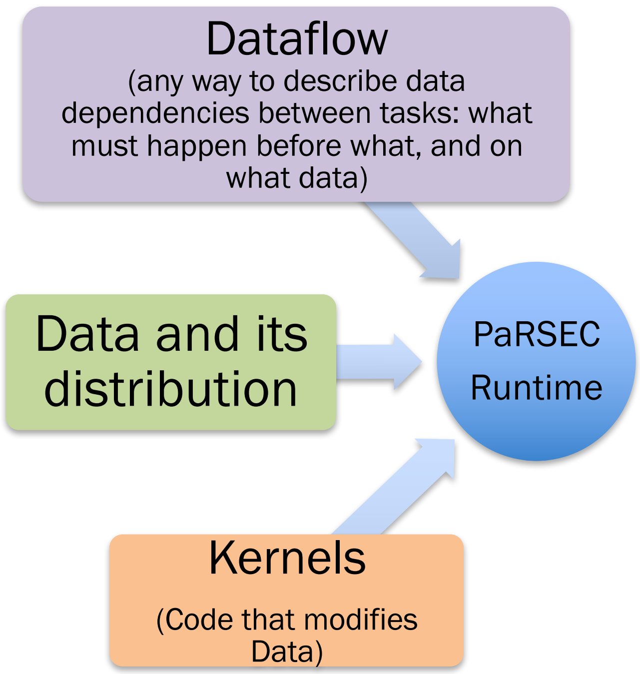
Task-based programming





- Focus on data dependencies, data flows, and tasks
- Don't develop for an architecture but for a portability layer
- Let the runtime deal with the hardware characteristics
  - But provide as much user control as possible
- StarSS, StarPU, Swift, Parallex, Quark, Kaapi, DuctTeip, and PaRSEC

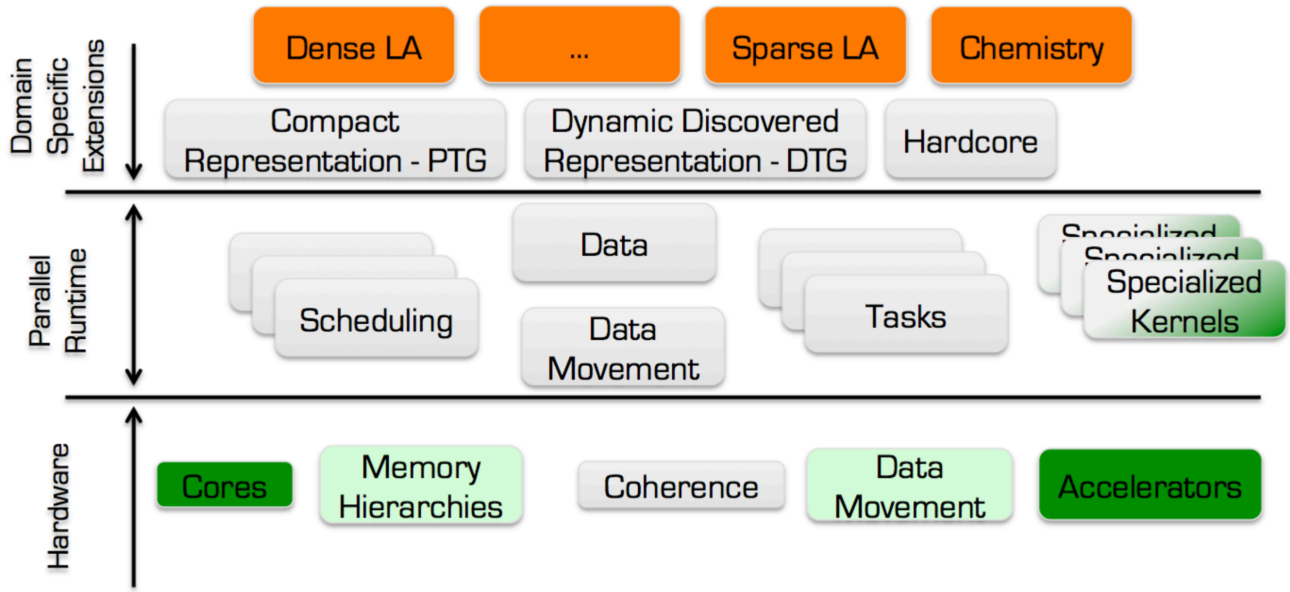
PaRSEC Runtime System Inputs



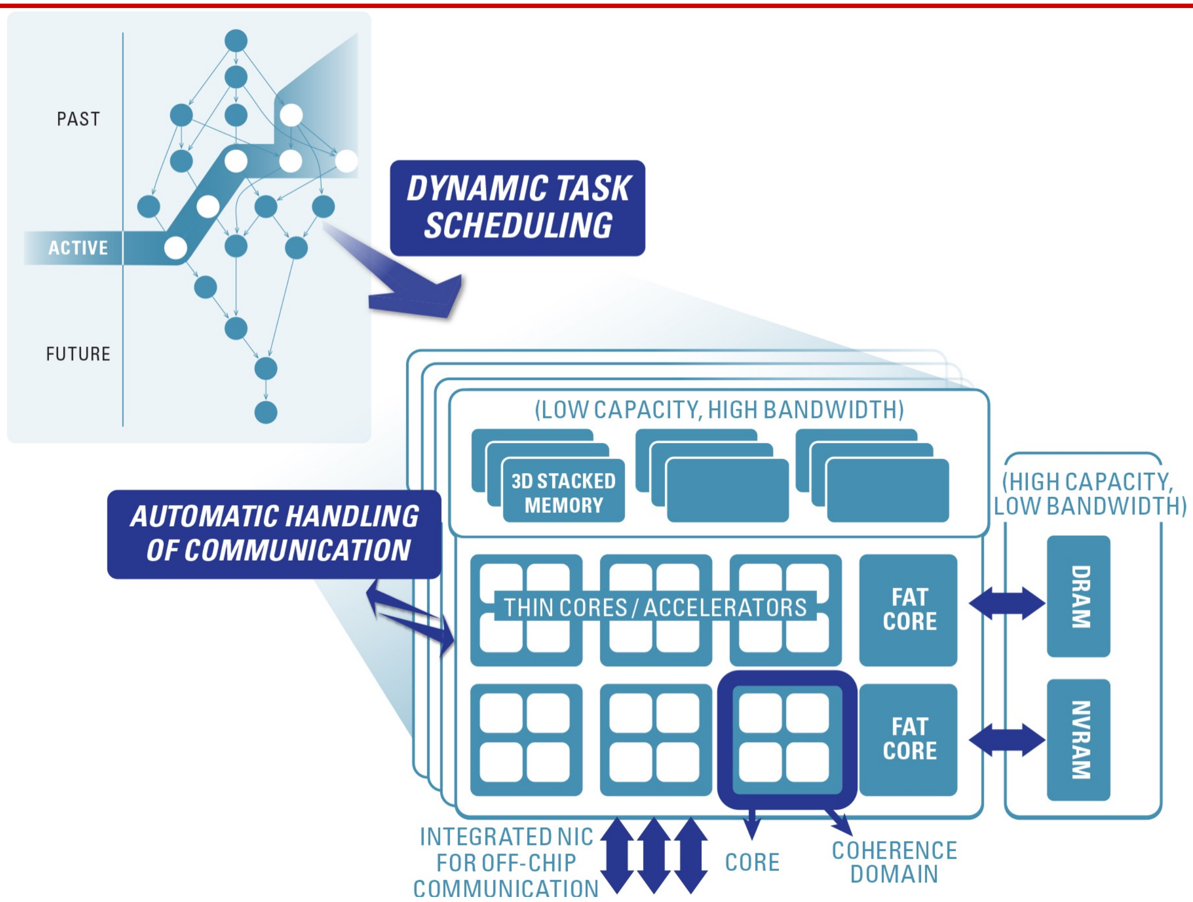
- Runtime System:
  - Manages local parallelism
  - Schedules tasks on cores and on accelerators
  - Manages memory
  - Adapts the execution to the local hardware (NUMA)
  - Moves data between nodes transparently and asynchronously

Dataflow with Runtime Scheduling

divide and orchestrate



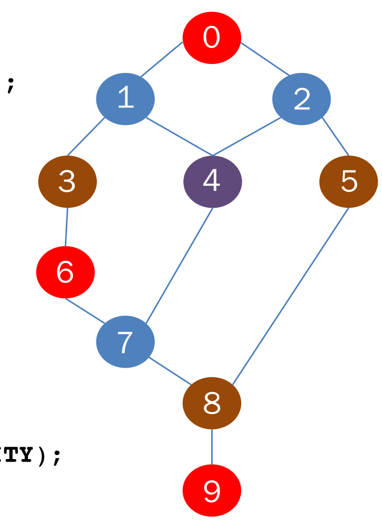
At the Node Level



Cholesky Factorization (3x3)

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

```
for( k = 0; k < total; k++ ) {

    parsec_insert_task( "Potrf", TILE_OF(A, k, k), INOUT | AFFINITY);

    for( m = k+1; m < total; m++ ) {
        parsec_insert_task( "Trsm",
                             TILE_OF(A, k, k), INPUT,
                             TILE_OF(A, m, k), INOUT | AFFINITY);

    }
    for( m = k+1; m < total; m++ ) {
        parsec_insert_task( "Herk",
                            TILE_OF(A, m, k), INPUT,
                            TILE_OF(A, m, m), INOUT | AFFINITY);

        for( n = m+1; n < total; n++ ) {
            parsec_insert_task( "Gemm",
                                 TILE_OF(A, n, k), INPUT,
                                 TILE_OF(A, m, k), INPUT,
                                 TILE_OF(A, n, m), INOUT | AFFINITY);

        }
    }
}
```



Inserting 1 x POTRF

```
for( k = 0; k < total; k++ ) {
    parsec_insert_task( "Potrf", TILE_OF(A, k, k), INOUT | AFFINITY);
    for( m = k+1; m < total; m++ ) {
        parsec_insert_task( "Trsm",
                            TILE_OF(A, k, k), INPUT,
                            TILE_OF(A, m, k), INOUT | AFFINITY);

    }
    for( m = k+1; m < total; m++ ) {
        parsec_insert_task( "Herk",
                           TILE_OF(A, m, k), INPUT,
                           TILE_OF(A, m, m), INOUT | AFFINITY);

        for( n = m+1; n < total; n++ ) {
            parsec_insert_task( "Gemm",
                               TILE_OF(A, n, k), INPUT,
                               TILE_OF(A, m, k), INPUT,
                               TILE_OF(A, n, m), INOUT | AFFINITY);

        }
    }
}
```

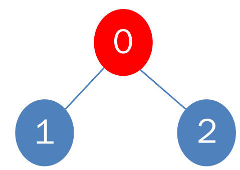
0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

0

Inserting 2 x TRSM

```
for( k = 0; k < total; k++ ) {

    parsec_insert_task( "Potrf", TILE_OF(A, k, k), INOUT | AFFINITY);

    for( m = k+1; m < total; m++ ) {
        parsec_insert_task( "Trsm",
                                 TILE_OF(A, k, k), INPUT,
                                 TILE_OF(A, m, k), INOUT | AFFINITY);

    }
    for( m = k+1; m < total; m++ ) {
        parsec_insert_task( "Herk",
                                TILE_OF(A, m, k), INPUT,
                                TILE_OF(A, m, m), INOUT | AFFINITY);

        for( n = m+1; n < total; n++ ) {
            parsec_insert_task( "Gemm",
                                    TILE_OF(A, n, k), INPUT,
                                    TILE_OF(A, m, k), INPUT,
                                    TILE_OF(A, n, m), INOUT | AFFINITY);

        }
    }
}
```

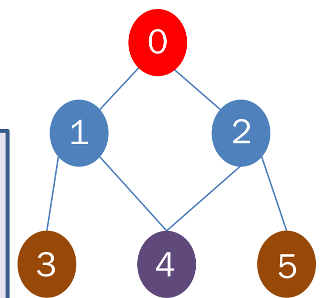
0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2



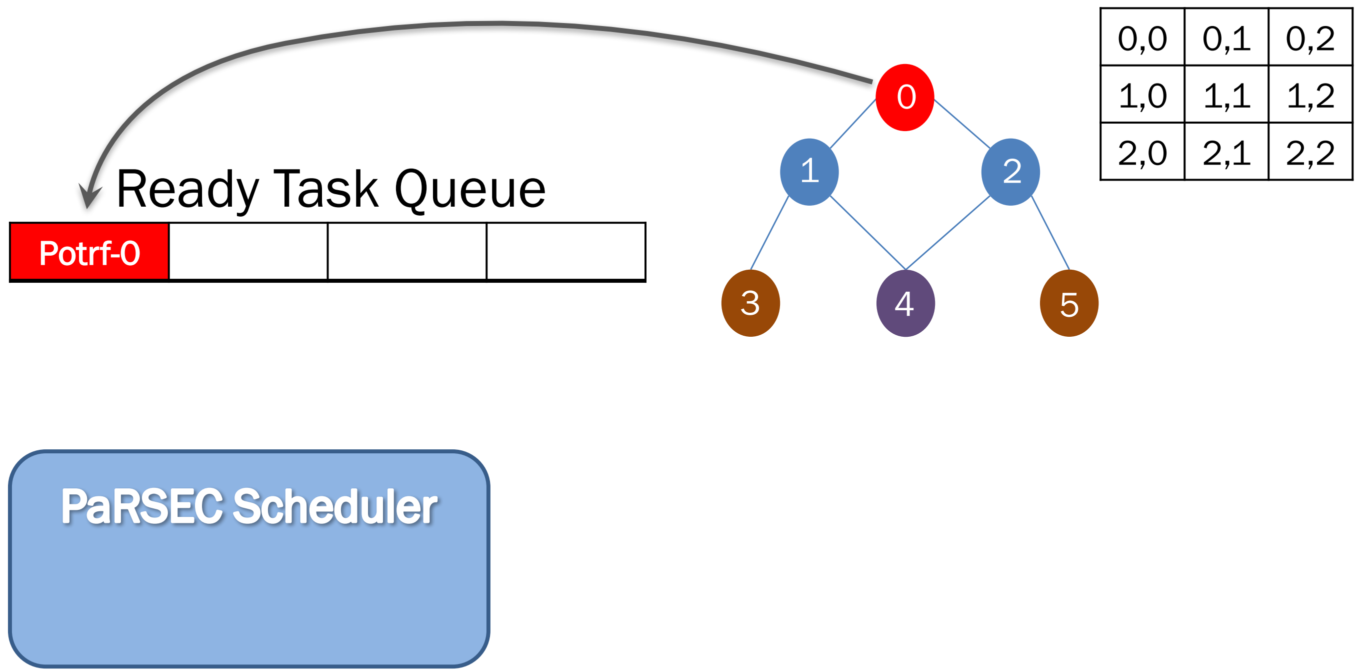
Inserting 2 x HERK + 1 x GEMM

```
for( k = 0; k < total; k++ ) {

    parsec_insert_task( "Potrf", TILE_OF(A, k, k), INOUT | AFFINITY);

    for( m = k+1; m < total; m++ ) {
        parsec_insert_task( "Trsm",
                                     TILE_OF(A, k, k), INPUT,
                                     TILE_OF(A, m, k), INOUT | AFFINITY);

    }
    for( m = k+1; m < total; m++ ) {
        parsec_insert_task( "Herk",
                                    TILE_OF(A, m, k), INPUT,
                                    TILE_OF(A, m, m), INOUT | AFFINITY);

        for( n = m+1; n < total; n++ ) {
            parsec_insert_task( "Gemm",
                                        TILE_OF(A, n, k), INPUT,
                                        TILE_OF(A, m, k), INPUT,
                                        TILE_OF(A, n, m), INOUT | AFFINITY);

        }
    }
}
```

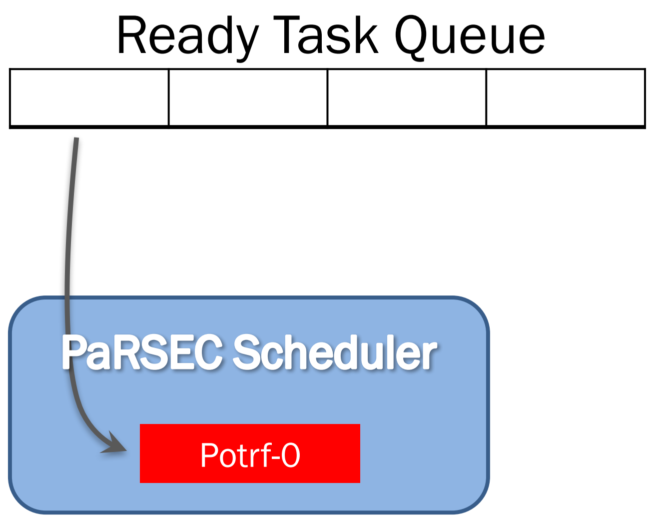
0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

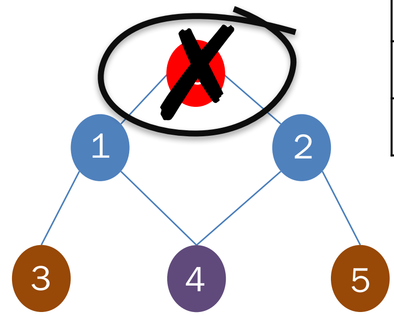


POTRF is ready to execute



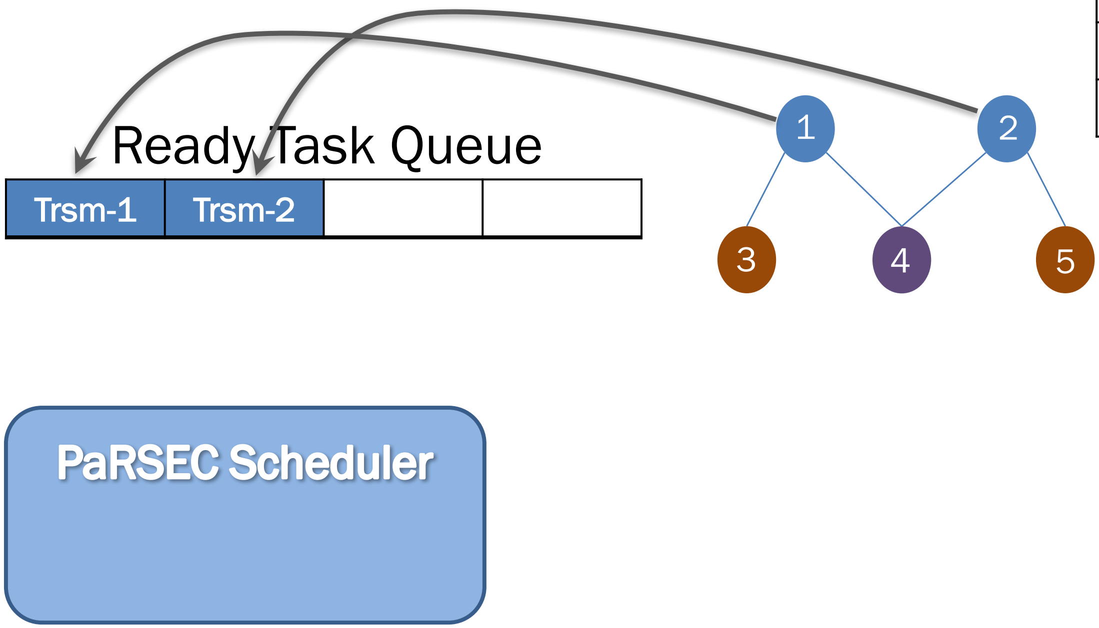
POTRF executes & leaves DAG





0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

TRSM tasks become ready

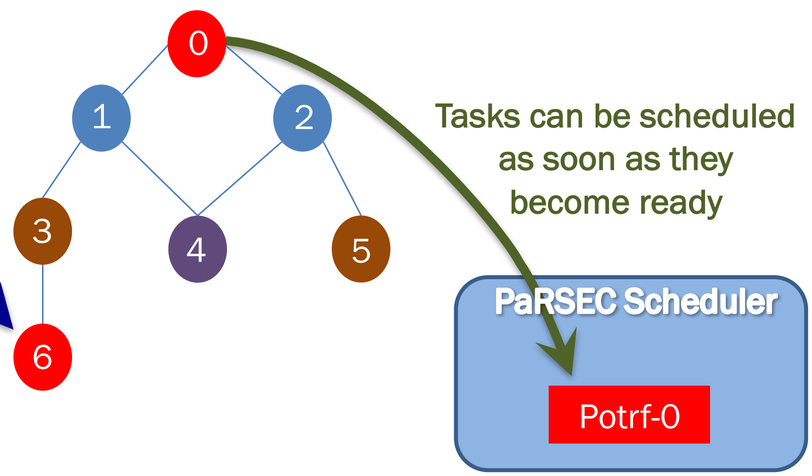


0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

Superscalar task execution

PaRSEC does not wait for the whole DAG to be built
before initiating the execution of tasks.

Tasks are inserted in the
DAG as soon as
they are discovered



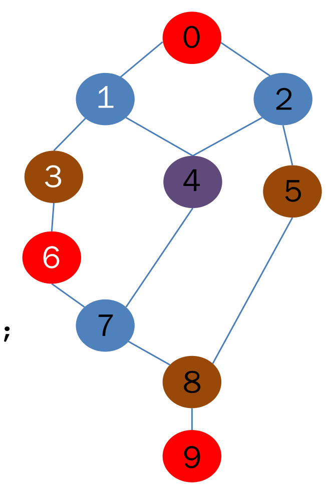

```
for( k = 0; k < total; k++ ) {

    parsec_insert_task( "Potrf", TILE_OF(A, k, k), INOUT | AFFINITY);

    for( m = k+1; m < total; m++ ) {
        parsec_insert_task( "Trsm",
                            TILE_OF(A, k, k), INPUT,
                            TILE_OF(A, m, k), INOUT | AFFINITY);

    }
    for( m = k+1; m < total; m++ ) {
        parsec_insert_task( "Herk",
                           TILE_OF(A, m, k), INPUT,
                           TILE_OF(A, m, m), INOUT | AFFINITY);

        for( n = m+1; n < total; n++ ) {
            parsec_insert_task( "Gemm",
                                TILE_OF(A, n, k), INPUT,
                                TILE_OF(A, m, k), INPUT,
                                TILE_OF(A, n, m), INOUT | AFFINITY);

        }
    }
}
```

Multi-node & Data placement

```
for( k = 0; k < total; k++ ) {

    parsec_insert_task( "Potrf", TILE_OF(A, k, k), INOUT | AFFINITY);

    for( m = k+1; m < total; m++ ) {
        parsec_insert_task( "Trsm",
                             TILE_OF(A, k, k), INPUT,
                             TILE_OF(A, m, k), INOUT | AFFINITY);

    }
    for( m = k+1; m < total; m++ ) {
        parsec_insert_task( "Herk",
                            TILE_OF(A, m, k), INPUT,
                            TILE_OF(A, m, m), INOUT | AFFINITY);

        for( n = m+1; n < total; n++ ) {
            parsec_insert_task( "Gemm",
                                 TILE_OF(A, n, k), INPUT,
                                 TILE_OF(A, m, k), INPUT,
                                 TILE_OF(A, n, m), INOUT | AFFINITY);

        }
    }
}
```

Data distribution between nodes (user defined)

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

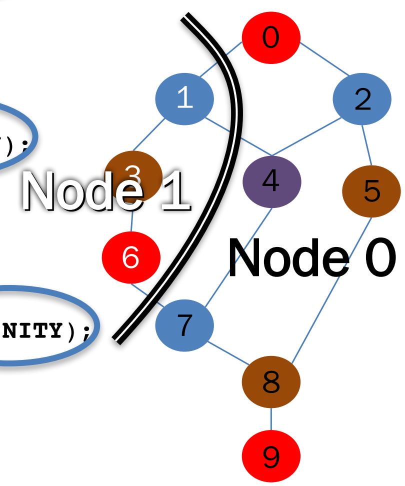


Task affinity follows data placement

```
for( k = 0; k < total; k++ ) {

    parsec_insert_task( "Potrf", TILE_OF(A, k, k), INOUT | AFFINITY);

    for( m = k+1; m < total; m++ ) {
        parsec_insert_task( "Trsm",
                            TILE_OF(A, k, k), INPUT,
                            TILE_OF(A, m, k), INOUT | AFFINITY);

    }
    for( m = k+1; m < total; m++ ) {
        parsec_insert_task( "Herk",
                           TILE_OF(A, m, k), INPUT,
                           TILE_OF(A, m, m), INOUT | AFFINITY);

        for( n = m+1; n < total; n++ ) {
            parsec_insert_task( "Gemm",
                                TILE_OF(A, n, k), INPUT,
                                TILE_OF(A, m, k), INPUT,
                                TILE_OF(A, n, m), INOUT | AFFINITY);

        }
    }
}
```

Data distribution between nodes (user defined)

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

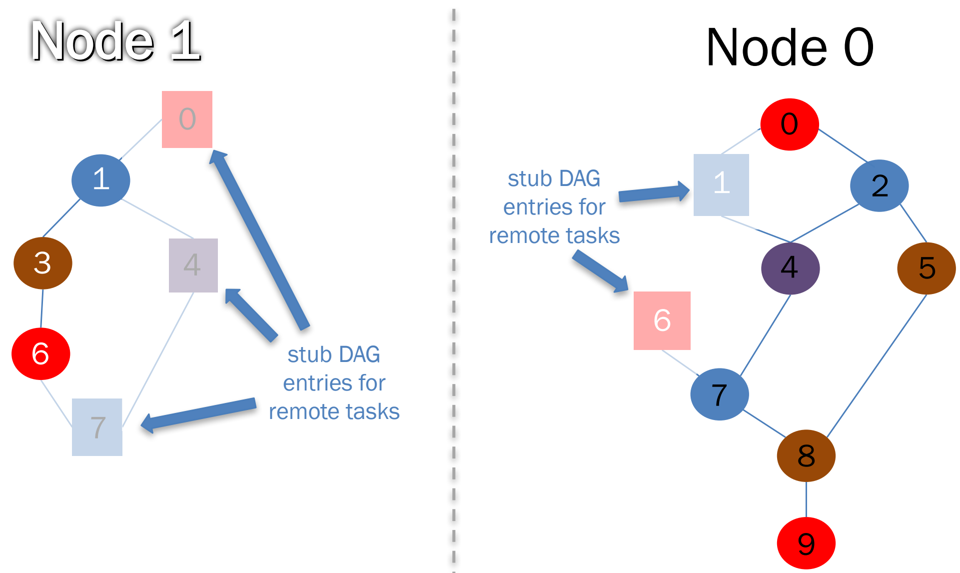


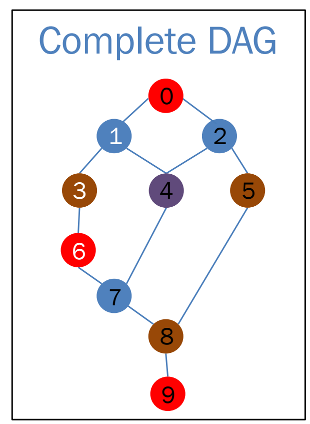
DAG building in distributed memory

On a given node, PaRSEC ignores remote tasks, except for immediate predecessors and children of local tasks.

Data distribution between nodes (user defined)

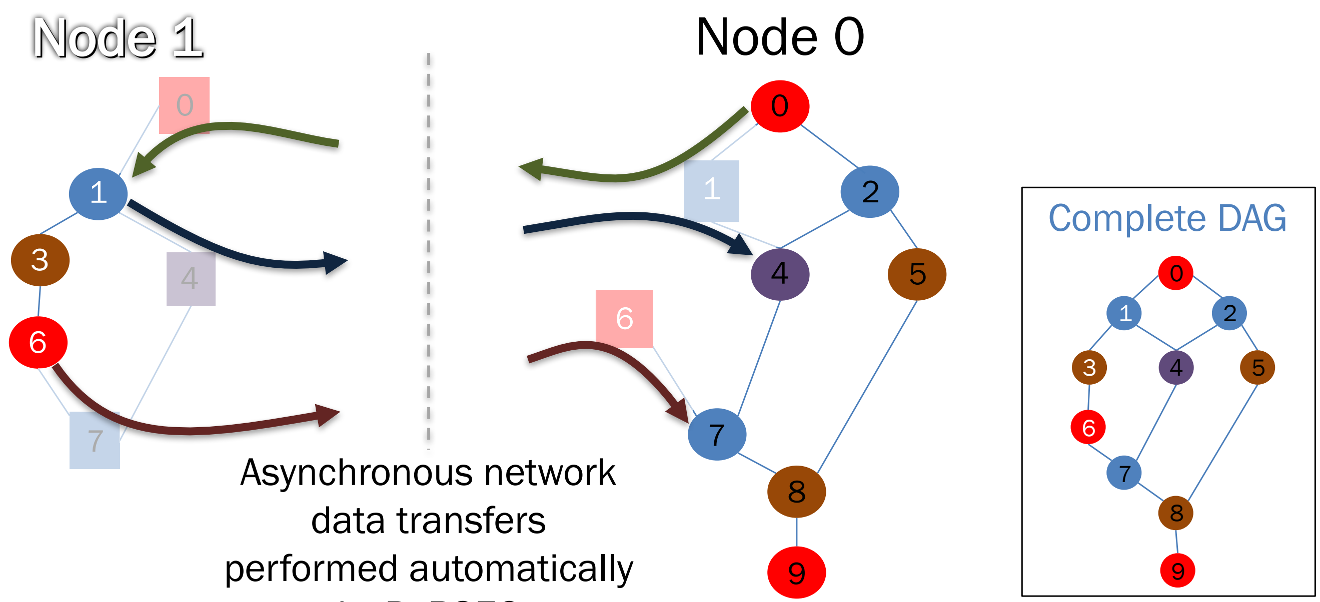
0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2





Implicit message passing

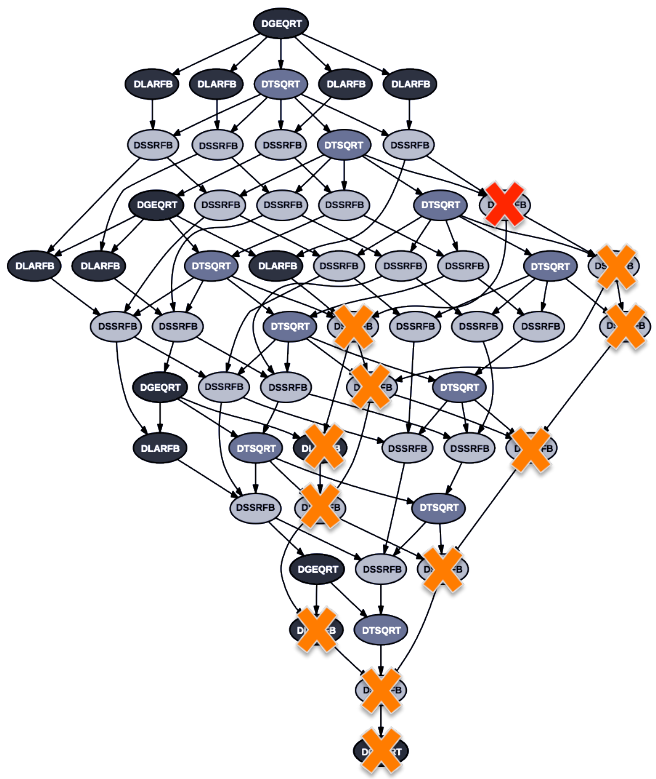
PaRSEC knows which node to exchange data with (due to the stubs) and does so <u>without user involvement</u>.



Asynchronous network data transfers performed automatically by PaRSEC

Resilience

automatic error recovery



- A fault propagates in the system according to data dependencies.
- If the original data can be recovered,

  automatic fault recovery is possible.

SLATE Features

- Runtime interface
  - Use Open-MP
  - Be able to plug into other systems
    - PaRSEC, Legion, Darma, StarPU, …
    - Statically scheduled on across nodes; dynamically schedule within node
- Tiled Algorithms
  - Runtime scheduling based on dataflow
  - Runtime dependency tracking
    - Plug into the different runtime systems
- Data distribution as in ScaLAPACK
  - Given the layout and arrangement of processes communication is understood
- Task based parallelism as in PLASMA
  - DAG based to allow overlap of computation and communication
- Ability to use accelerators as in MAGMA
  - Hybrid computing using the runtime system

Today: Integration with DOE ECP Applications

- **Underdevelopment and design**
- **xSDK – Coordination of NLA libraries across DOE**
- **PEEKS – Iterative methods**
- **Working with the ECP applications, i.e. Chemistry: diagonalization**
- **Link seamlessly and work efficiently when used as LAPACK and/or ScaLAPACK replacement**
- **European Project**
  - NLAFET H2020