MULTIGRID SOLVER

Iterative solver

- Consider the system Au =f and let v be an approximation of u

- There are two important measures
- The error e = u – v
- The residual r = f – Av

- With norm $\|e\|_2 = \sqrt{\sum_{i=1}^{N} e_i^2}$

- It follows: Ae=Au – Av = f – Av = r

Residual Equation:

$$Ae = r$$

Residual Correction:

$$u = v + e$$

Relaxation scheme : Jacobi

- A = L+D+U
- Au = f becomes (L+D+U) u =f
- Du =-(L+U)u +f

$$u = -D^{-1}(L+U)u + D^{-1}f$$

Let $R_J = -D^{-1}(L+U)$, then the iteration is:

$$u = R_J u + D^{-1}f$$

$$v^{(new)} = R_J v^{(old)} + D^{-1}f$$

Relaxation scheme: Gauss seidel

- A= L+D+U
- Au =f becomes (L+D+U)u = f
- (L+D)u = -Uu + f

$$u = -(D-L)^{-1}Uu + (D-L)^{-1}f$$

Let $R_G = -(D-L)^{-1}U$

Then iterate $v^{(new)} \leftarrow R_G v^{(old)} + (D-L)^{-1}f$

Error propagation: $e^{(new)} \leftarrow R_G e^{(old)}$

Convergence

- e (new) = R e(old)
- After m iterations
- em = R^M e0
- The iterative scheme converge if
-

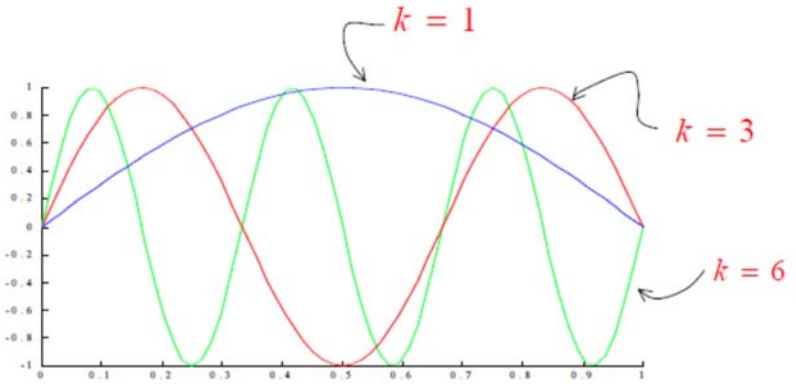
$$\rho(R) < 1$$

Numerical Experiments

Solve $Au = 0$, $-u_{i-1} + 2u_i - u_{i+1} = 0$

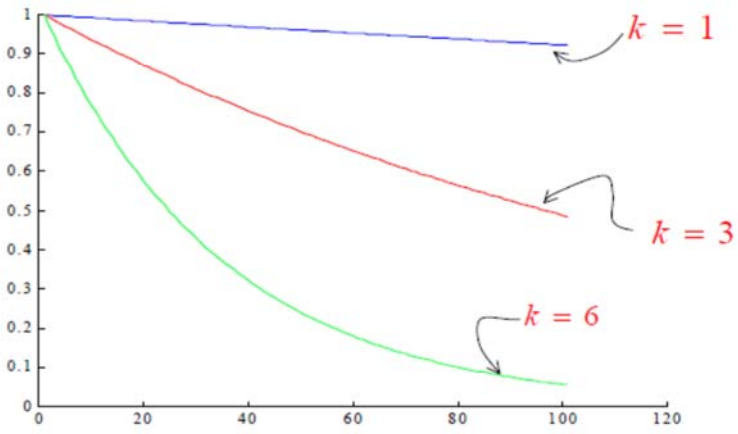
Use Fourier modes as initial iterate, with N=64:

$$\vec{v_k} = (v_i)_k = \sin\left(\frac{ik\pi}{N}\right) \qquad \underset{\text{component}}{1 \le i \le N-1}, \quad \underset{\text{mode}}{1 \le k \le N-1}$$



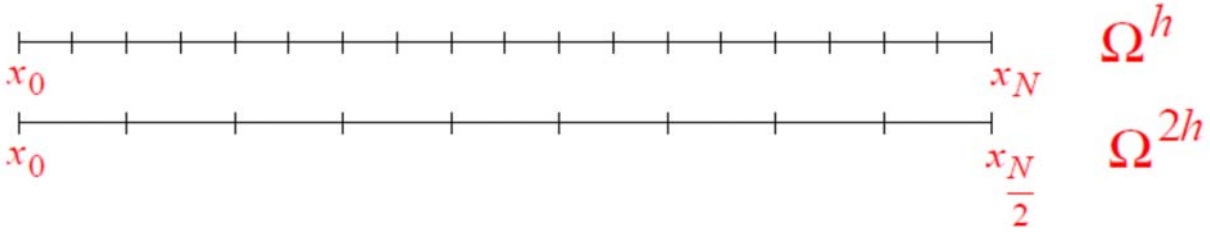
Convergence rates differ for different error components

Error, $||e||_\infty$, in weighted Jacobi on Au = 0 for 100 iterations using initial guesses of v_1, v_3, and v_6



First observation toward multigrid

- Many relaxation schemes have the smoothing property, where oscillatory modes of the error are eliminated effectively, but smooth modes are damped very slowly.
- This might seem like a limitation, but by using coarse grids we can use the smoothing property to good advantage.



Reason #1 for using coarse grids: Nested Iteration

- Coarse grids can be used to compute an improved initial guess for the fine-grid relaxation. This is advantageous because:
  - Relaxation on the coarse-grid is much cheaper (1/2 as many points in 1D, 1/4 in 2D, 1/8 in 3D)
  - Relaxation on the coarse grid has a marginally better convergence rate, for example

$$1 - O(4h^2) \qquad \text{instead of} \qquad 1 - O(h^2)$$

Second observation toward multigrid:

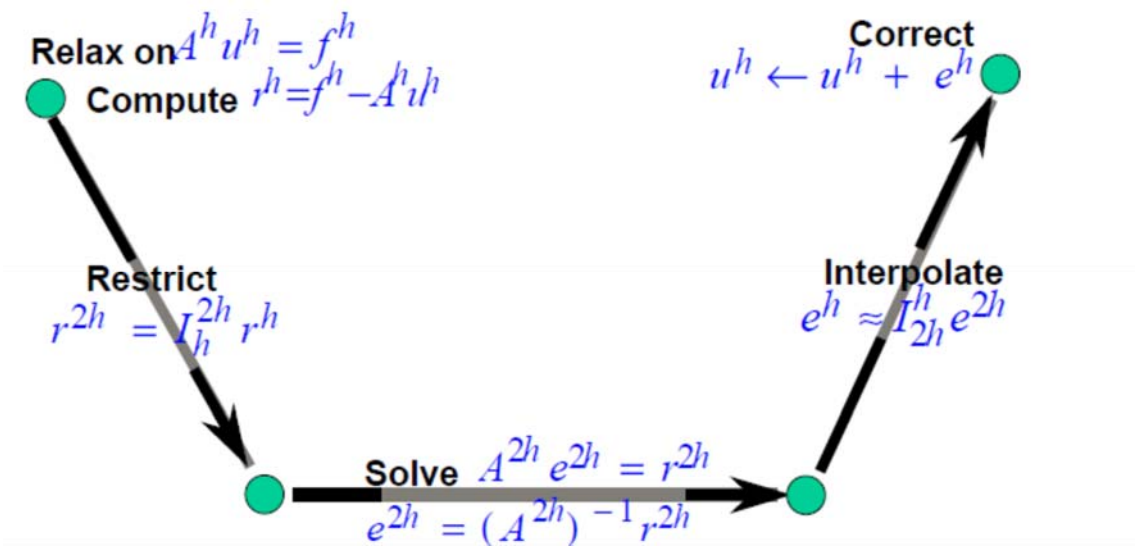
- Recall the residual correction idea: Let v be an approximation to the solution of Au=f, where the residual r=f-Av. The the error e=u-v satisfies Ae=r.
- After relaxing on Au=f on the fine grid, the error will be smooth. On the coarse grid, however, this error appears more oscillatory, and relaxation will be more effective.
- Therefore we go to a coarse grid and relax on the residual equation Ae=r, with an initial guess of e=0.

Coarse-grid correction

- Relax on Au=f on Ω^h to obtain an approximation v^h
- Compute $r = f - Av^h$.
- Relax on Ae=r on Ω^{2h} to obtain an approximation to the error, e^{2h}.
- Correct the approximation $v^h \leftarrow v^h + e^{2h}$.
- Clearly, we need methods for the mappings

$\Omega^h \Rightarrow \Omega^{2h}$ and $\Omega^{2h} \Rightarrow \Omega^h$

Coarse-grid Correction



$$A = A_h = \textit{original matrix}$$
$$R = R_h^{2h} = \textit{restriction matrix}$$
$$I = I_{2h}^h = \textit{interpolation matrix}.$$

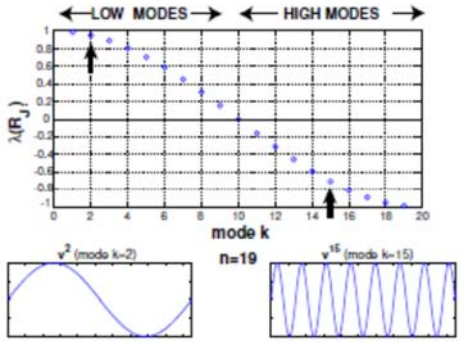
- **R = transpose I**

$$\textit{The coarse grid matrix is } A_{2h} = R_h^{2h} A_h I_{2h}^h = RAI$$

Smoother

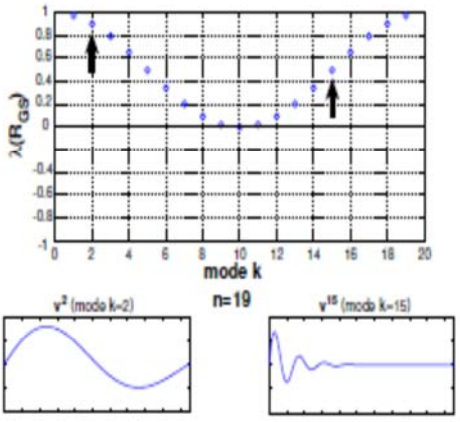
If the *high frequency* components of the error decay faster than the *low frequency* components, we say that the iterative method is a *smoother*.

2.2.1 Jacobi



Is Jacobi a smoother? $\ldots \rightarrow NO$

- GAUS SEIDEL



Is Gauss-Seidel a good smoother? $\ldots$

2 level theory

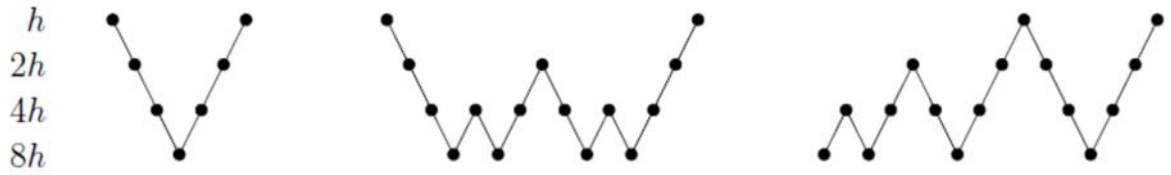
- Fine grid: $r = A e$
- Restrict: $RA e$
- Solve on coarse grid: $(RAI)^{-1} RA e$
- Interpolate: $I (RAI)^{-1} RA e$

- $E= I (RAI)^{-1} RA e$ $S=I (RAI)^{-1} RA$ $E=Se$
- $S^2= I (RAI)^{-1} RA \ I \ (RAI)^{-1} RA= I (RAI)^{-1} RA=S$
- S has eigenvalues 0,1

- Multigrid error E is not the full error but a projection of e

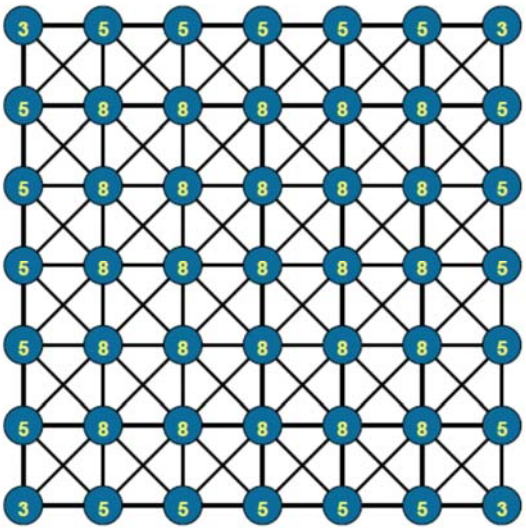
- Iterations handles the high frequencies.
- Multigrid handles the low frequenccies

Extend 2 levels to multi levels

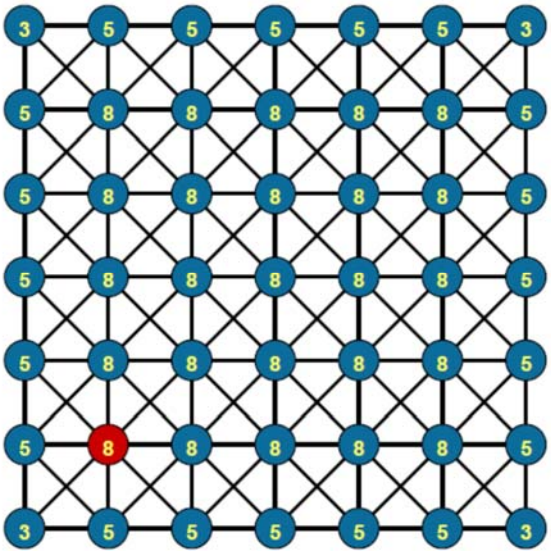


V-cycles and W-cycles and FMG use several grids several times.

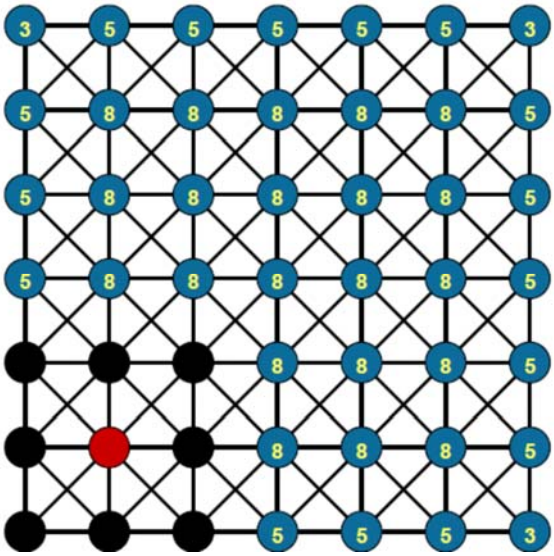
Coarsening: RS algorithm



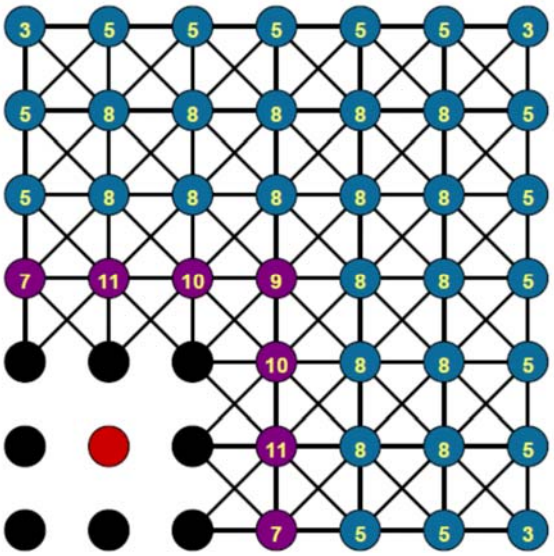
- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of F-pt neighbors



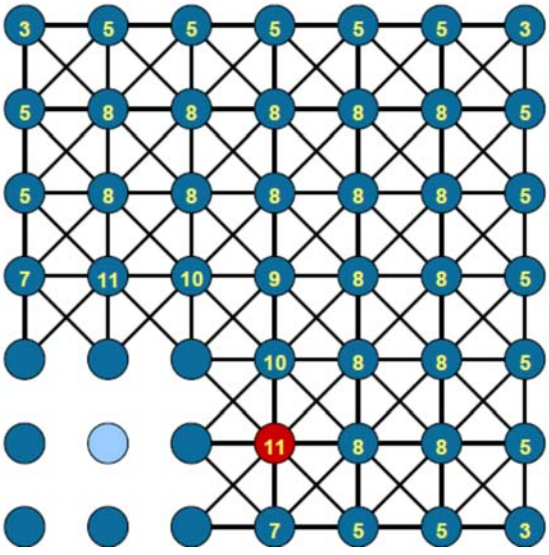
- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of F-pt neighbors



- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of F-pt neighbors



- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of F-pt neighbors



- select C-pt with maximal measure
- select neighbors as F-pts
- update measures of F-pt neighbors

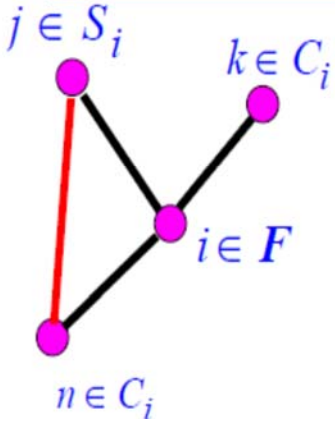
- Strength of connection: Given a threshold $0 < \theta \leq 1$, we say that variable u_i strongly depends on variable u_j if

$$-a_{ij} \geq \theta \max_{k \neq i} \{-a_{ik}\}$$

- The basic coarsening procedure is as follows:
  - Define a strength matrix A_s by deleting weak connections in A
  - First pass: Choose an independent set of fine-grid points based on the graph of A_s
  - Second pass: Choose additional points if needed to satisfy interpolation requirements

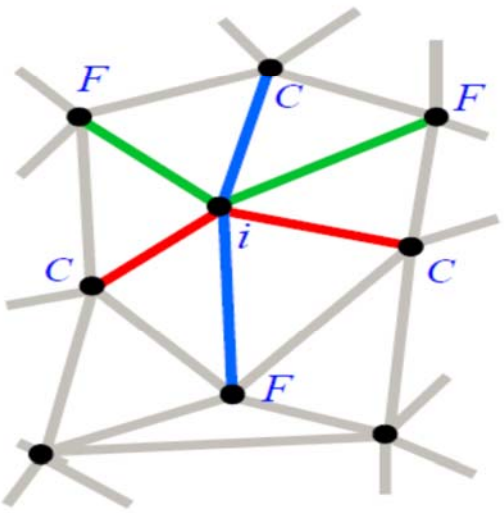
- First Criterion: F - F dependence

— (C1) For each $i \in F$, each point $j \in S_i$ should either be in C itself or should depend on at least one point in C_i.



Since the value of u_i depends on the value of u_j, the value of u_j must be represented on the coarse-grid for good interpolation. If j isn't a C-point, it should depend on a point in C_i so its value is "represented" in the interpolation.

To define prolongation at i, we must examine the types of connections of u_i.



Sets of connection types:

C_i — *i* is dependent on these coarse interpolatory C-points.

D_i^s — *i* is dependent on these F-points.

D_i^w — i does not depend on these "weakly connected" points, which may be C- or F-points.

$$a_{ii}e_i = -\sum_{j\in C_i} a_{ij}e_j - \sum_{j\in F_i^s} a_{ij}e_j - \sum_{j\in N_i^w} a_{ij}e_j$$

C_i : C-points strongly connected to i
F_i^s : F-points strongly connected to i
N_i^w : all points weakly connected to i.

- Write

$$e_j = \begin{cases} \sum_{k\in C_i}\left(\dfrac{a_{jk}}{\sum_{l\in C_i} a_{jl}}\right) e_k, & j \in F_i^s \\ e_i, & j \in N_i^w \end{cases}$$

- Then

$$w_{ij} = -\left(a_{ij} + \sum_{k\in F_i^s}\left(\frac{a_{kj}}{\sum_{l\in C_i} a_{kl}}\right) a_{ik}\right) \Big/ \left(a_{ii} + \sum_{k\in N_i^w} a_{ik}\right)$$

MULTIGRID IMPLEMENTATION

Coarsening

- Problem 1: take the maximum set
- Strategy: maintain a shorter list

- Problem 2: update the measure
- Strategy: use radix 2 linear merge

Build interpolation

- Problem1: matching 2 rows
- Strategy: linear merge

- Problem2: summing
- Strategy:
- avoid division when only one C point match for the F point
- keep track of the sum

Build restriction

- Problem: take a transpose of sparse array
- Strategy:
- count the elements for rows on the new transpose and get the shape
- sequential scan of original array and fill each element at corresponding row with incremental counter

Build coarse array

- Problem: 2 sparse matrix multiply
- Strategy:
- with RAI start with AI product using radix 2 linear merge
- row i of A points to the rows of I for K-merge
- exploit 1.d0 element of I to reduce the size of the K-merge

Gauss seidel

- Unroll dot product by 4
- Combine Residual calculation with last gauss seidel iteration lower part is identical
- Combine Residual norm with last gauss iteration

Multigrid solver

- Rescale the array with 1.d0 diagonal
- Get the number of levels so to get a small array at the lowest grid level
- Solve that array with direct sparse solver
- Use F-AMG to get cheap good start guess
- Keep track of residual on both relaxation and multigrid to exit at tolerance level

	Nov.04	25-Jul
Problem name	BiCG	AMG
※ NOTE		coarsetype=1
VFS7_Conduction	146.6	**141.3**
VFT6_SEP_1D	36.5	**44.1**
VFT6_SEP_2D_STR	78.6	**141.8**
VFT6_SEP_2D_UNS	1405.8	**1751.4**
VFT8_Dam_2D	6.4	**7.6**
VFT9_AW_3D_UNS	56.8	**71.4**
VFS2_wall_motion	4.0	**4.4**
VFS3_Duct_flow(STR)	2.4	**2.2**
VFS3_Duct_flow(UNS)	9.3	**8.3**
VFT1_2D_boiling	4.6	**4.9**
VD10_PSBT	5486.9	**5346.8**
VFS9_Blockage	6.8	**6.9**
VFS10_Annulus	4.3	**4.4**
VD9_STERN	908.4	**941.6**
VD7_PASCAL	649.5	**760.0**
VFS8_Turbulent_flow	245.4	**279.4**
VFS8_Turbulent_ke_3D	89.4	**35.3**
VFS12_Boron	79.8	**65.0**
VD3_SUBO	1319.8	**1804.5**
VD4_Bankoff	3849.8	**4058.8**
VD5_ROCOM	211.3	**242.4**
VFT9_AirWater_tetra	351.8	**672.6**
VFT3_flashing	15.5	**10.3**
VFT4_cavitation	143.6	**161.3**
VFT5_gas_plume	88.2	**68.6**
VFT7_manometric	58.8	**38.6**
VFT10_Blowdown	521.3	X
VD6_fluidic_device	3038.4	**2605.4**
4c (100x100x100)	1859.0	**1922.6**
2D Ductflow (nn=100,000)	1065.5	**368.9**
2D Ductflow (nn=1,000,000)		

INTEL XEON PHI KNL

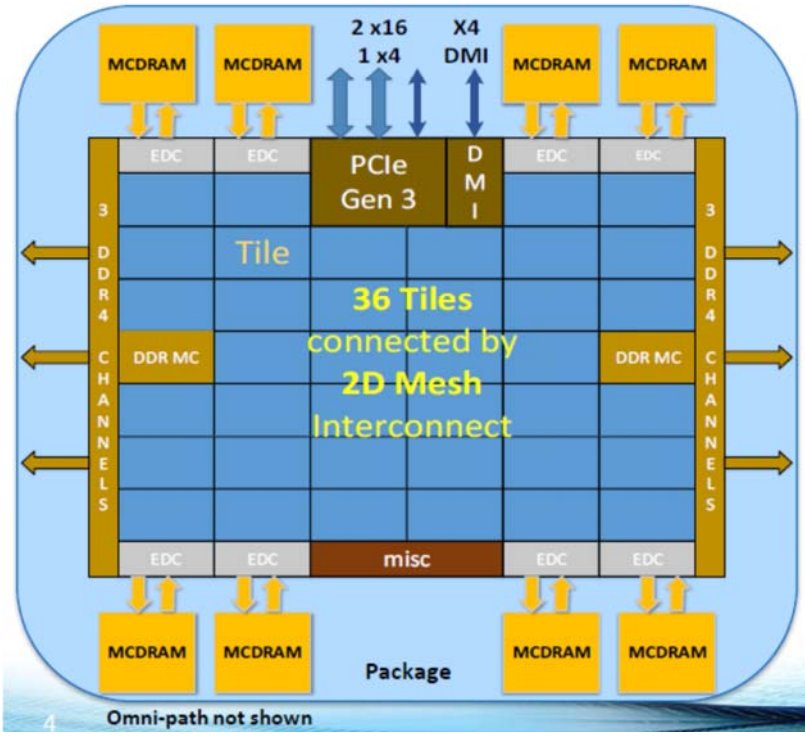
First **self-boot** Intel® Xeon Phi™ processor that is **binary compatible** with main line IA. Boots standard OS.

Significant improvement in scalar and **vector** performance

Integration of **Memory on package**: innovative memory architecture for high bandwidth and high capacity

Integration of **Fabric on package**

Knights Landing Overview



Omni-path not shown

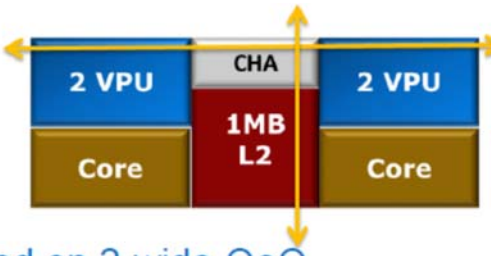
Chip: 36 Tiles interconnected by **2D Mesh**
Tile: 2 Cores + 2 VPU/core + 1 MB L2

Memory: MCDRAM: 16 GB on-package; High BW
DDR4: 6 channels @ 2400 up to 384GB
IO: 36 lanes PCIe Gen3. 4 lanes of DMI for chipset
Node: 1-Socket only
Fabric: Omni-Path on-package (not shown)

Vector Peak Perf: 3+TF DP and 6+TF SP Flops
Scalar Perf: ~3x over Knights Corner
Streams Triad (GB/s): MCDRAM : 400+; DDR: 90+

Source Intel: All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice. KNL data are preliminary based on current expectations and are subject to change without notice. 1Binary Compatible with Intel Xeon processors using Haswell Instruction Set (except TSX). 2Bandwidth numbers are based on STREAM-like memory access pattern when MCDRAM used as flat memory. Results have been estimated based on internal Intel analysis and [illegible] purposes only. Any difference in system [illegible] scalar performance.

KNL Tile: 2 Cores, each with 2 VPU
1M L2 shared between two Cores



Core: Changed from Knights Corner (KNC) to KNL. Based on 2-wide OoO Silvermont™ Microarchitecture, but with *many* changes for HPC.

4 thread/core. Deeper OoO. Better RAS. Higher bandwidth. Larger TLBs.

2 VPU: 2x AVX512 units. 32SP/16DP per unit. X87, SSE, AVX1, AVX2 and EMU

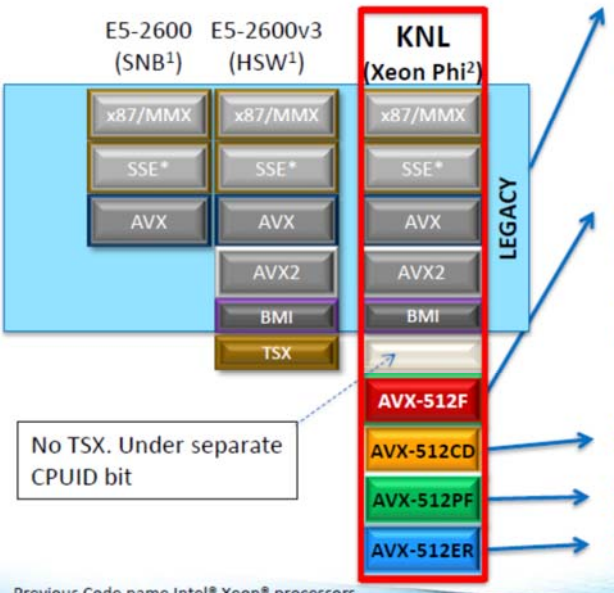
L2: 1MB 16-way. 1 Line Read and ½ Line Write per cycle. Coherent across all Tiles

CHA: **C**aching/**H**ome **A**gent. Distributed Tag Directory to keep L2s coherent. MESIF protocol. 2D-Mesh connections for Tile

Many Trailblazing Improvements in KNL

Improvements	What/Why
Self Boot Processor	No PCIe bottleneck
Binary Compatibility with Xeon	Runs all legacy software. No recompilation.
New Core: Atom™ based	~3x higher ST performance over KNC
Improved Vector density	3+ TFLOPS (DP) peak per chip
New AVX 512 ISA	New 512-bit Vector ISA with Masks
Scatter/Gather Engine	Hardware support for gather and scatter
New memory technology: MCDRAM + DDR	Large High Bandwidth Memory → MCDRAM Huge bulk memory → DDR
New on-die interconnect: Mesh	High BW connection between cores and memory
Integrated Fabric: Omni-Path	Better scalability to large systems. Lower Cost

KNL ISA



KNL implements all legacy instructions

- Legacy binary runs w/o recompilation
- KNC binary requires recompilation

KNL introduces AVX-512 Extensions

- 512-bit FP/Integer Vectors
- 32 registers, & 8 mask registers
- Gather/Scatter

Conflict Detection: Improves Vectorization

Prefetch: Gather and Scatter Prefetch

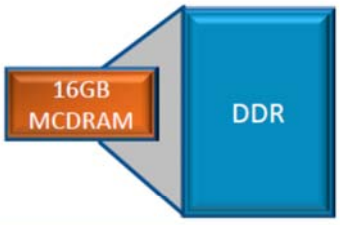
Exponential and Reciprocal Instructions

1. Previous Code name Intel® Xeon® processors
2. Xeon Phi = Intel® Xeon Phi™ processor

Memory Modes

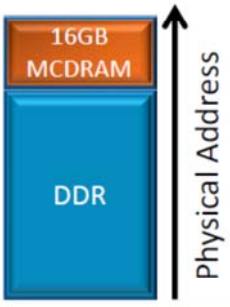
Three Modes. Selected at boot

Cache Mode



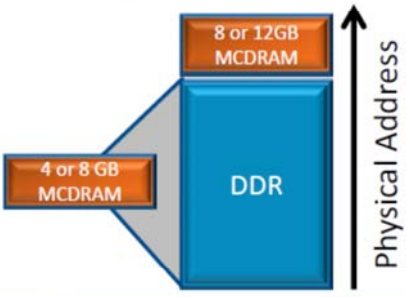
- SW-Transparent, Mem-side cache
- Direct mapped. 64B lines.
- Tags part of line
- Covers whole DDR range

Flat Mode



- MCDRAM as regular memory
- SW-Managed
- Same address space

Hybrid Mode



- Part cache, Part memory
- 25% or 50% cache
- Benefits of both

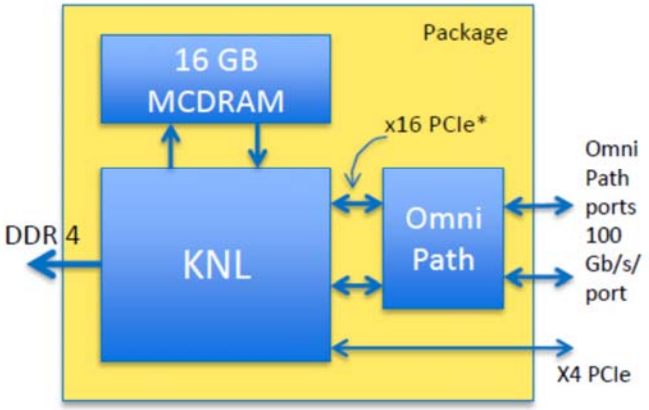
KNL with Omni-Path™

Omni-Path™ Fabric integrated *on package*

First product with integrated fabric

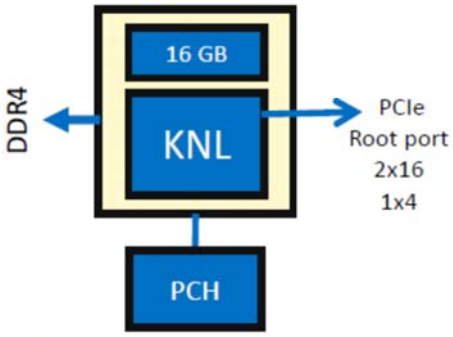
Connected to KNL die via 2 x16 PCIe* ports
Output: 2 Omni-Path ports
- 25 GB/s/port (bi-dir)

Benefits
- Lower cost, latency and power
- Higher density and bandwidth
- Higher scalability



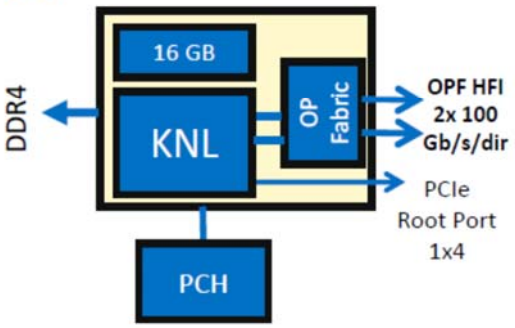
*On package connect with PCIe semantics, with MCP optimizations for physical layer

Knights Landing Products



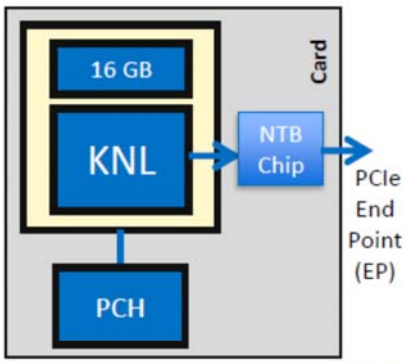
KNL

DDR Channels: 6
MCDRAM: up to 16 GB
Gen3 PCIe (Root port): 36 lanes



KNL with Omni-Path

DDR Channels: 6
MCDRAM: up to 16 GB
Gen3 PCIe (Root port): 4 lanes
Omni-Path Fabric: 200 Gb/s/dir



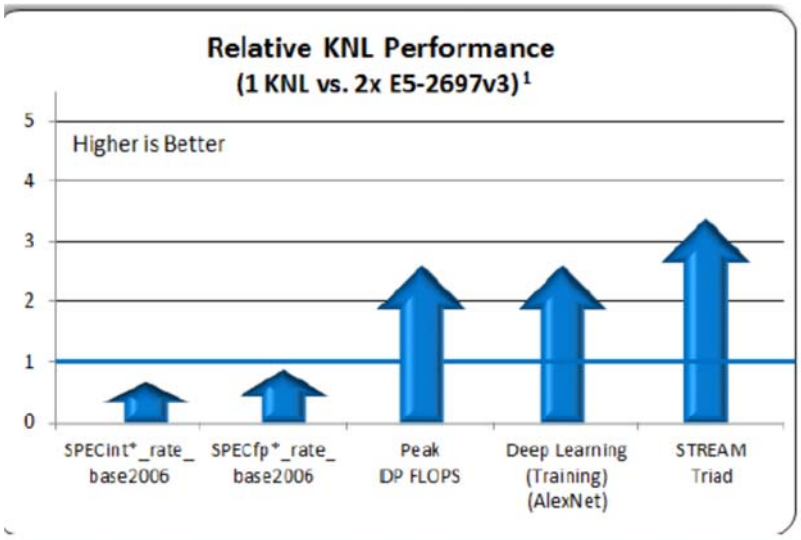
KNL Card

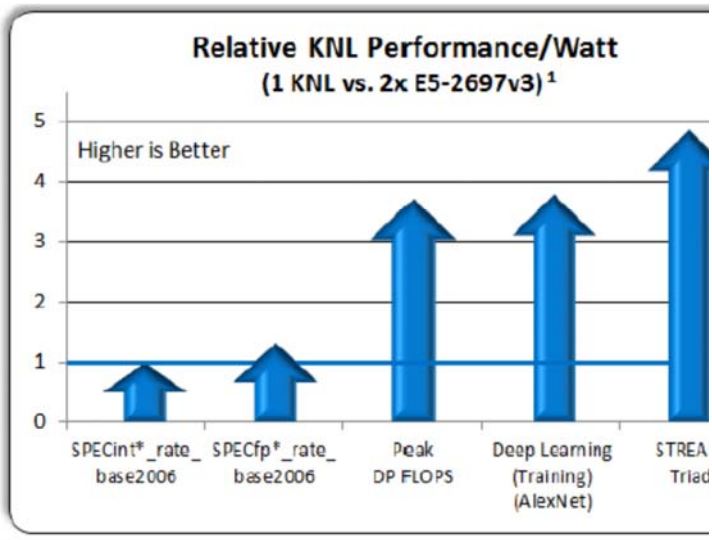
No DDR Channels
MCDRAM: up to 16 GB
Gen3 PCIe (End point): 16 lanes
NTB Chip to create PCIe EP

Self Boot Socket

PCIe Card

Potential future options subject to change without notice. Codenames.
All timeframes, features, products and dates are preliminary forecasts and subject to change without further





Significant performance improvement for compute and bandwidth sensitive workloads, while still providing good general purpose throughput performance.

1. Projected KNL Performance (1 socket, 200W CPU TDP) vs. 2 Socket Intel® Xeon® processor E5-2697v3 (2x145W CPU TDP)

ftware and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer stems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your ntemplated purchases, including the performance of that product when combined with other products. Source E5-2697v3: www.spec.org. Intel measured AlexNet images/sec. All results have been estimated based on ernal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance. For more information go to: tp://www.intel.com/performance *Other names and brands may be claimed as the property of others.

KNL MODELS

- 7210 1.3Ghz DDR4-2133 64 cores
- 7230 1.3Ghz DDR4-2400 64 cores
- 7250 1.4Ghz DDR4-2400 68 cores
- 7290 1.5Ghz DDR4-2400 72 cores
- All 14nm
-
- 2018 KNC 10nm more cores =>100

MANUFACTURERS

- Supermicro
- 1 node Workstation
- Rack 2U with 4 KNL nodes and local disks

- FUJI
- Rack 2U with 8 KNL nodes diskless