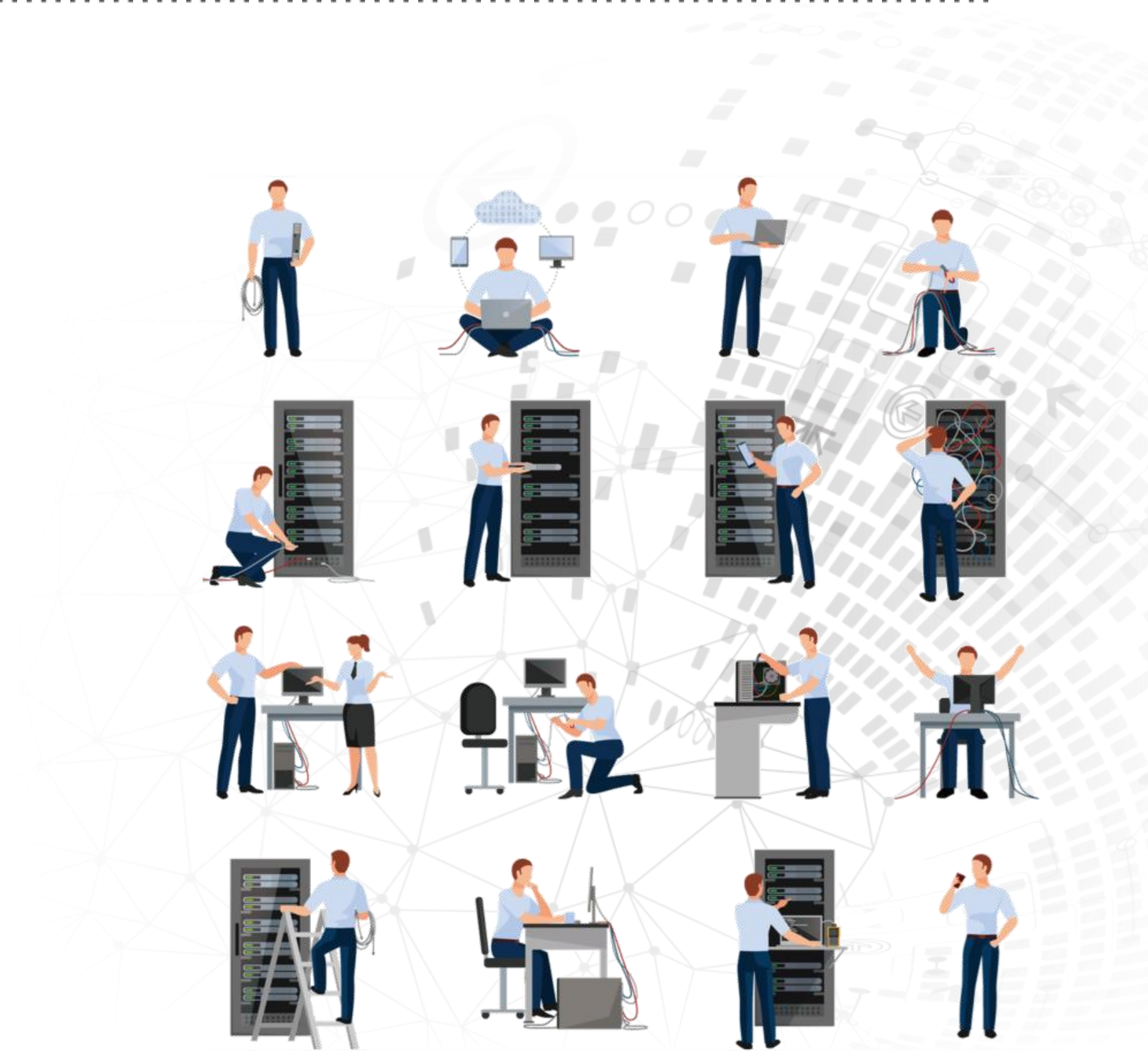


# 04 클러스터 테스트

성능 테스트



## 4. 테스트

Category	Program	Requirement	Target system
Component	Stream	Mandatory <sup>2)</sup>	CPU partition and GPU partition
	OSU Micro-Benchmarks	Mandatory	GPU partition
	IOR	Mandatory	GPU partition
	MDTEST	Mandatory	GPU partition
	Stride	Reference <sup>3)</sup>	CPU partition and GPU partition
	HPCC RandomAccess	Reference	CPU partition and GPU partition
	CLOMP	Reference	CPU partition and GPU partition
Comprehensive (Standard)	HPL	Mandatory	CPU partition and GPU partition
	HPL-AI	Mandatory	GPU partition
	HPCG	Mandatory	CPU partition and GPU partition
	GREEN500	Mandatory	GPU partition
	GRAPH500	Mandatory	CPU partition or GPU partition
	MLPerf	Mandatory	GPU partition
	P3DFFT	Mandatory	CPU partition or GPU partition
Application	LAMMPS	Mandatory	GPU partition
	PELE-LM	Mandatory	CPU partition and GPU partition <sup>4)</sup>
	PySCF	Mandatory	CPU partition
	Quantum Espresso	Mandatory	GPU partition
	QUEST	Mandatory	CPU partition or GPU partition or memory partition

## 4. 테스트

Value of MKL_ENABLE_INSTRUCTIONS	ISA
AVX512	Intel AVX-512 for systems based on Intel® Xeon® processors
AVX512_MIC	Intel AVX-512 for systems based on Intel® Xeon Phi™ processors and coprocessors
AVX2	Intel AVX2
AVX	Intel AVX
SSE4_2	Intel SSE4-2

- To turn on automatic CPU-based dispatching of Intel AVX-512 on systems based on Intel Xeon processors:

For the bash shell:

**export MKL\_ENABLE\_INSTRUCTIONS=AVX512**

For a C shell (csh or tcsh):

**setenv MKL\_ENABLE\_INSTRUCTIONS AVX512**

- To configure the library not to dispatch more recent architectures than Intel AVX2:

For the bash shell:

**export MKL\_ENABLE\_INSTRUCTIONS=AVX2**

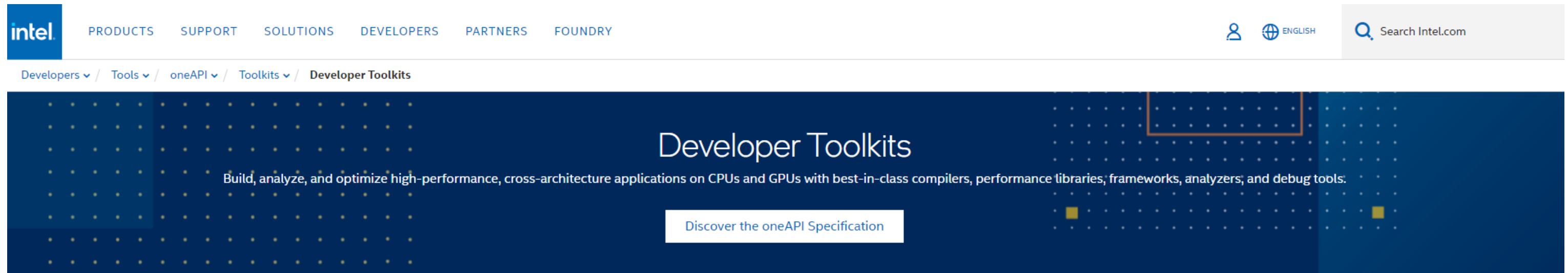
For a C shell (csh or tcsh):

**setenv MKL\_ENABLE\_INSTRUCTIONS AVX2**

## 4. 테스트

### HPL

<https://www.intel.com/content/www/us/en/developer/tools/oneapi/toolkits.html>



### Select Your Toolkit

Download what you need for any project.

#### Intel® oneAPI Base Toolkit

Develop performant, data-centric applications across Intel® CPUs and GPUs with this foundational toolset.

##### General Compute

- Intel® oneAPI DPC++/C++ Compiler
- Intel® DPC++ Compatibility Tool
- Intel® Distribution for GDB\*
- Intel® oneAPI DPC++ Library (oneDPL)
- Intel® oneAPI Threading Building Blocks (oneTBB)
- Intel® oneAPI Math Kernel Library (oneMKL)
- Intel® oneAPI Deep Neural Networks Library (oneDNN)
- Intel® oneAPI Data Analytics Library (oneDAL)
- Intel® oneAPI Collective Communications Library (oneCCL)
- Intel® Integrated Performance Primitives (Intel® IPP)
- Intel® Cryptography Primitives Library

#### Intel® oneAPI HPC Toolkit

Build, analyze, and scale HPC applications across shared and distributed memory computing systems.

##### High-Performance Computing

- Intel oneAPI DPC++/C++ Compiler
- Intel® Fortran Compiler
- Intel DPC++ Compatibility Tool
- Intel Distribution for GDB
- Intel oneAPI Threading Building Blocks (oneTBB)
- Intel oneAPI DPC++ Library (oneDPL)
- Intel® MPI Library
- Intel oneAPI Math Kernel Library (oneMKL)
- Intel oneAPI Deep Neural Networks Library (oneDNN)
- Intel oneAPI Data Analytics Library (oneDAL)
- Intel oneAPI Collective Communications Library (oneCCL)

#### AI Frameworks & Tools

Accelerate end-to-end data science and machine learning pipelines using Python\* tools and frameworks.

##### End-to-End AI and Machine Learning Acceleration

- Python 3.9
- Intel® Extension for PyTorch\* (CPU)
- Intel Extension for PyTorch (GPU)
- Intel® Extension for TensorFlow\* (CPU)
- Intel Extension for TensorFlow (GPU)
- Intel® Optimization for XGBoost\*
- Intel® Extension for Scikit-learn\*
- Modin\*
- Intel® Neural Compressor

[Learn More](#)

## 4. 테스트

### HPL

High Performance Linpack (HPL)은  
선형 방정식  $Ax=b$ 의 해를 찾는 문제를  
수치적으로 풀어내는 벤치마크 테스트

$$A = \begin{bmatrix} 2 & 3 & 1 \\ 4 & 7 & 2 \\ 6 & 18 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

이때,  $X$ 를 구하는 문제를 푸는 것

## 4. 테스트

### HPL

High Performance Linpack (HPL)은  
선형 방정식  $Ax=b$ 의 해를 찾는 문제를  
수치적으로 풀어내는 벤치마크 테스트

- LU 분해 (with partial pivoting) 기법을 사용하여 dense matrix를 푼다.
- 수행 시간은 주어진  $N \times N$  크기의 dense 행렬에 대해 얼마나 빠르게 문제를 푸는지를 측정한다.
- 측정 단위는 GFLOPS 또는 TFLOPS (초당 부동소수점 연산 횟수)로 표시된다

## 4. 테스트

### HPL

#### 성능 측정 방식

계산량: 약  $\frac{2}{3}N^3$  FLOP

LU 분해에서의 계산량은 다음과 같이 정의

**Forward Elimination:**  $\frac{2}{3}N^3 + \frac{2}{3}N^2 + \frac{1}{6}N$

**Back Substitution:**  $N^2$

*HPL 성능 측정에서는 leading term 인  $\frac{2}{3}N^3$  만 사용*

왜냐하면:  $N^3$  이 매우 크기 때문에 나머지 항은 무시 가능한 수준의 연산량

예:  $N=100,000$  이면,

$$\frac{2}{3}N^3 = 6.67 \times 10^{14}$$

## 4. 테스트

### HPL

항목	설명
N	행렬 크기. 시스템 메모리에 맞게 최대화
NB	블록 사이즈. L2/L3 cache와 통신비 고려
P, Q	MPI 프로세서 그리드. 네트워크 트래픽 균형 조정
PMAP	Process Mapping (Row-major / Column-major)
alignment	행렬 정렬. SIMD 성능 영향

## 4. 테스트

### HPL 에서 말하는 N, NB

- N은 HPL 에 사용 하는 정방 행렬 A의 크기 ( $N \times N$ )
- NB (Block Size)는 HPL에서 사용하는 블록 분할 단위

N=2048, NB=256NB 이라면,

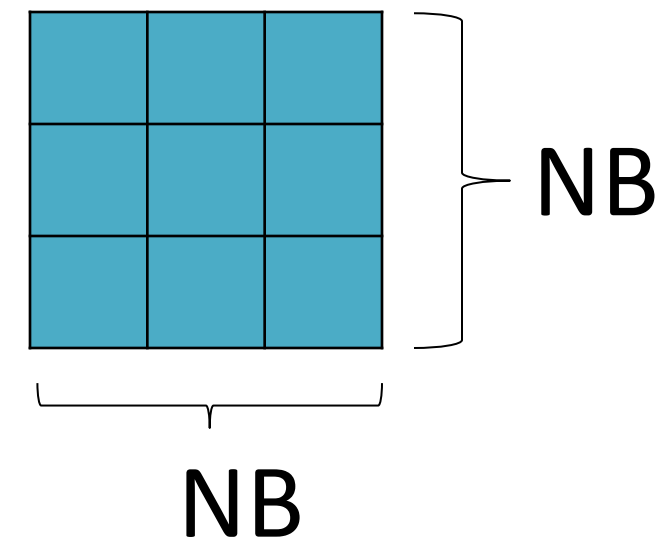
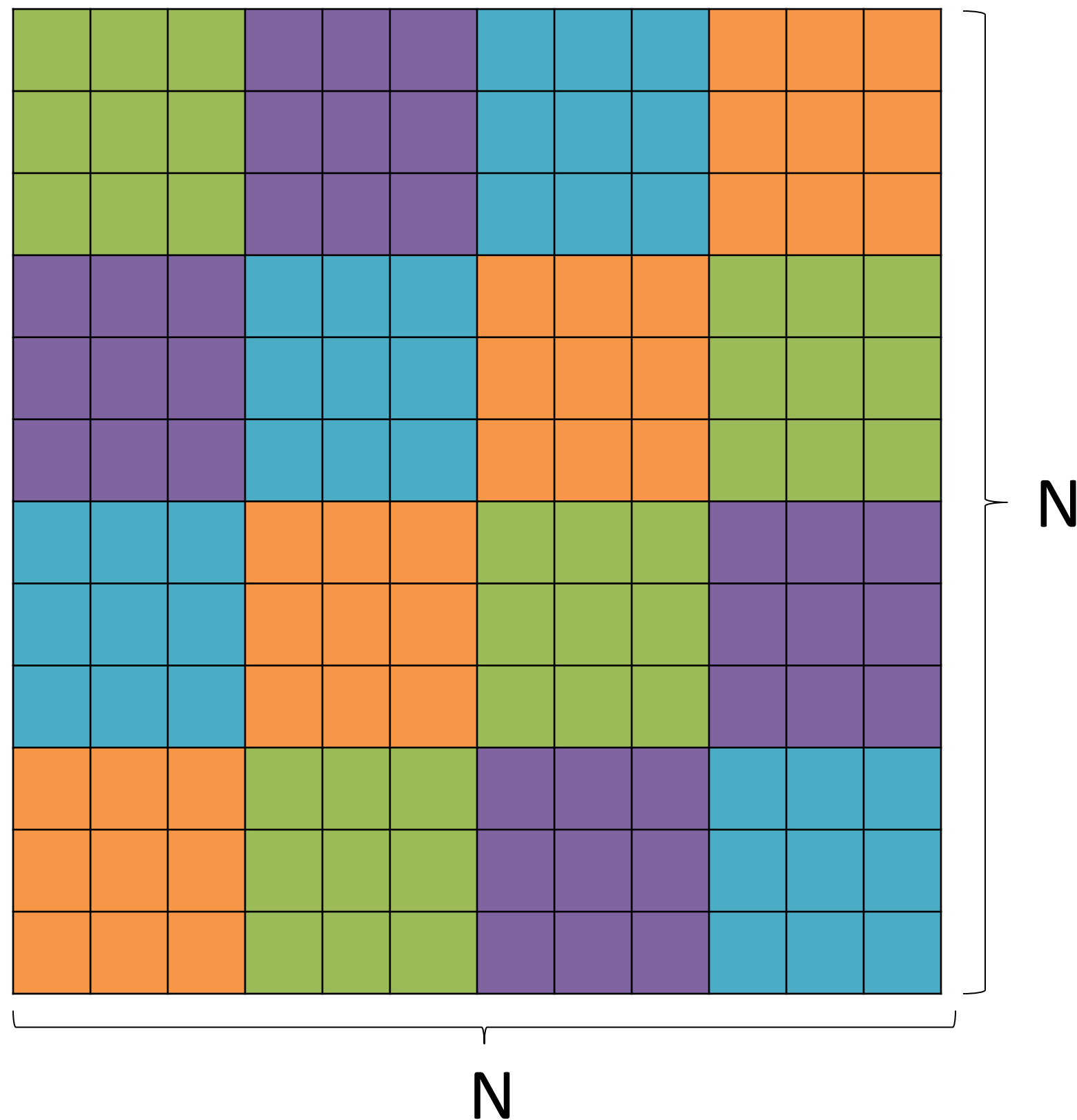
$\{N/NB\} \times \{N/NB\}$

$(2048/256)^2=64$  블록

전체 행렬은  $256 \times 256$ 크기의 작은 블록 64개로 나누어짐

## 4. 테스트

HPL 에서 말하는 N, NB



## 4. 테스트

### HPL 에서 말하는 N, NB 관계

- HPL은  $N \times N$  행렬을  $NB \times NB$  블록 단위로 나눔
- 만약  $N \bmod NB \neq 0$  이라면, **마지막 블록이 잘리게 되어** 계산과 통신이 복잡해짐

N	NB	블록 수	문제
128	32	$4 \times 4$ 블록	완벽히 맞음
128	30	$4.26 \times 4.26$ 블록	마지막 블록은 짜투리 영역 발생 → 불균형

### 계산 오버헤드 최소화

잘린 블록에서는 루프 언롤링, 벡터화, 캐시 프리패치 등의 최적화가 떨어짐  
불규칙한 크기의 연산은 내부 BLAS 라이브러리 성능을 저하시킵니다.

### MPI 통신 간소화

HPL은 2D block-cyclic 방식으로 데이터를 분산시키므로,  
NB 단위가 일정해야 모든 MPI 프로세스가 균형 있게 데이터를 나눠가질 수 있음  
마지막 줄/열의 블록이 불완전하면, 해당 프로세스는 계산량이 줄어 로드 밸런스가 깨짐

## 4. 테스트

### HPL 에서 말하는 N, NB 관계

조건	설명
$N \bmod NB = 0$	선호됨. 계산/통신 효율 높음
NB is multiple of cache line / SIMD width	캐시 최적화
NB matches optimized BLAS block size (e.g., 32, 64, 128)	내부 루틴과 일치
너무 크면	병렬성이 낮아지고 블록 수가 적어짐
너무 작으면	계산 단위 작아져 오버헤드 증가

벤치마크 수행 시 NB를 다양하게 실험하여 가장 높은 Rmax 값을 찾는 것이 일반적입니다.

## 4. 테스트

NB = 384

### AVX-512 구조에 맞춘 블록 정렬

- AVX-512는 512비트 SIMD 레지스터를 사용해, 8개의 64-bit doubles 또는 16개의 32-bit floats를 한 번에 처리할 수 있는 구조
- NB = 384는 64의 배수로,  $384 / 64 = 6 \Rightarrow$  곧 AVX-512 벡터 레지스터 6회 반복
- 이로 인해 DGEMM과 같은 내부 행렬 곱셈에서 벡터 사용이 최적화되어 부동소수점 처리량이 극대화

### 캐시 라인 정렬과 캐시 재사용 최적화

- CPU는 메모리에서 데이터를 64바이트 단위로 가져옴, 블록 하나를 캐시 라인(cache line)
- 한 행의 블록이  $384 = 384 \times 8 \text{ byte} = 3072 \text{ bytes}$ , 이는 48캐시 라인에 해당
- 1캐시 라인 = 64 byte
- 48 캐시 라인 =  $48 \times 64 = 3072 \text{ byte}$
- NB=384의 경우, 행 하나가 384개의 double(8바이트)을 담고 있으므로  $384 \times 8 = 3072$  바이트, 즉 48 캐시 라인과 동일  $\rightarrow$  각 블록을 접근할 때 **미스 없이 전체 캐시 라인을 효율적으로 사용**
- 캐시 라인에 오버랩 없이 정렬되어 데이터 접근 시 **효율이 높아짐**

## 4. 테스트

NB = 384

- NB = 256: AVX-512와 정렬은 되지만 너무 작아 벡터 재활용이 부족함
- NB = 512: AVX-512 정렬 가능하지만 캐시 사이즈 제한으로 **오버플로우** 발생

NB = 384는 중도 최적값으로, 캐시 & SIMD & 통신 오버헤드의 균형을 맞춘 설정

## 4. 테스트

P, Q

- HPL에서는 행렬을 2차원 block-cyclic 분할 방식으로 여러 노드/프로세스에 분산
- 이때 전체 MPI 프로세스를 P행 × Q열로 정렬하여 사용

총 MPI 프로세스 수: 64

가능 구성 예시:

P = 8, Q = 8

P = 4, Q = 16

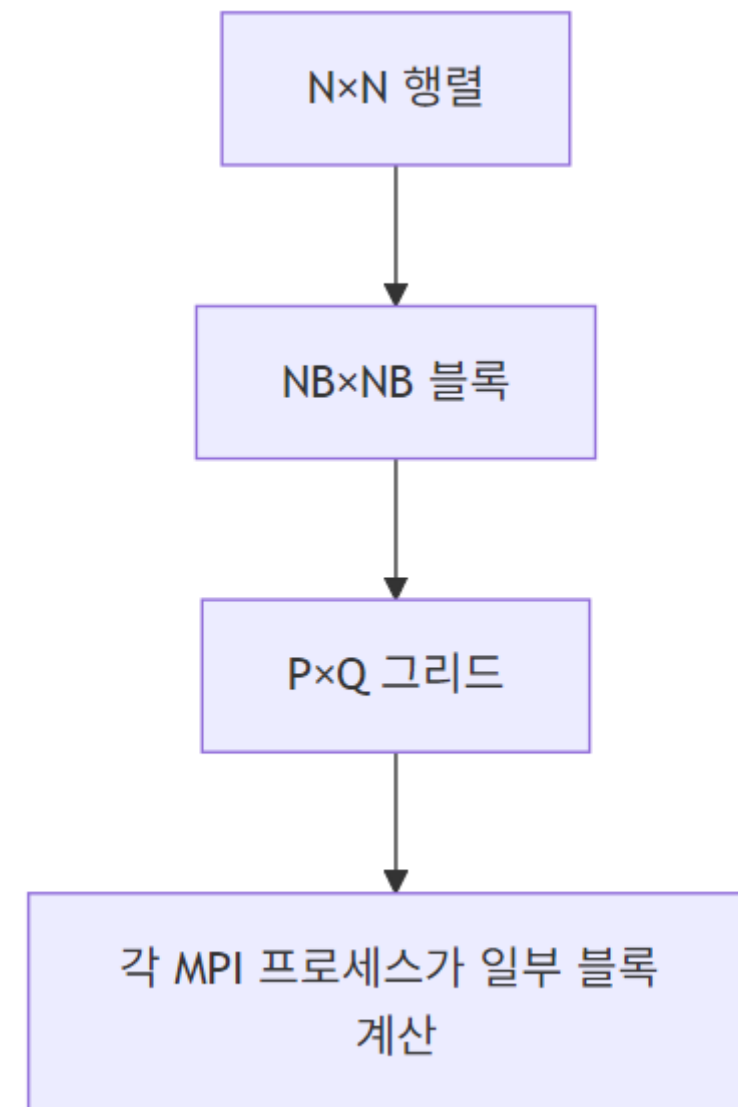
P = 2, Q = 32 등

(단,  $P \times Q$  = 총 프로세스 수를 반드시 만족)

## 4. 테스트

### MPI와 P, Q

- HPL은 각 블록을 block-cyclic 방식으로  $P \times Q$  구조에 따라 나누어 분배합니다:



## 4. 테스트

### MPI와 P, Q

- $P \times Q = \text{MPI 총 프로세스 수}$
- P - Q 가 작을수록 좋음 (즉 정사각형에 가까운 형태가 이상적)
- HPL 성능은 종종  $P < Q$ 에서 약간 더 좋음)

## 4. 테스트

### HPL 테스트 예 1.

- $N=1024$   $NB=128$   $P=4$   $Q=4$  node1 8개 코어 node2 8개 코어
- 총 행렬 크기:  $1024 \times 1024$ ,  $NB=128 \rightarrow$  총  $8 \times 8 = 64$  블록
- MPI 프로세스 수:  $P \times Q = 4 \times 4 = 16 \rightarrow$  노드당 8 MPI 프로세스
- 각 노드 환경: node1 (코어 0-7), node2 (코어 0-7)

#### 1단계: MPI 프로세스 분배

- MPI 총 프로세스 수 = 16
- 두 노드에 걸쳐 라운드로빈 방식 또는 코어당 1 프로세스 매핑

## 4. 테스트

### MPI와 P, Q

2단계:  $P \times Q = 4 \times 4$  프로세스 격자  
프로세스 랭크 → 노드와 코어 매핑 예

Rank	Node	Core
0	node1	0
1	node1	1
...	...	...
7	node1	7
8	node2	0
...	...	...
15	node2	7

## 4. 테스트

### MPI와 P, Q

#### 3단계: Block-Cyclic 분배 방식

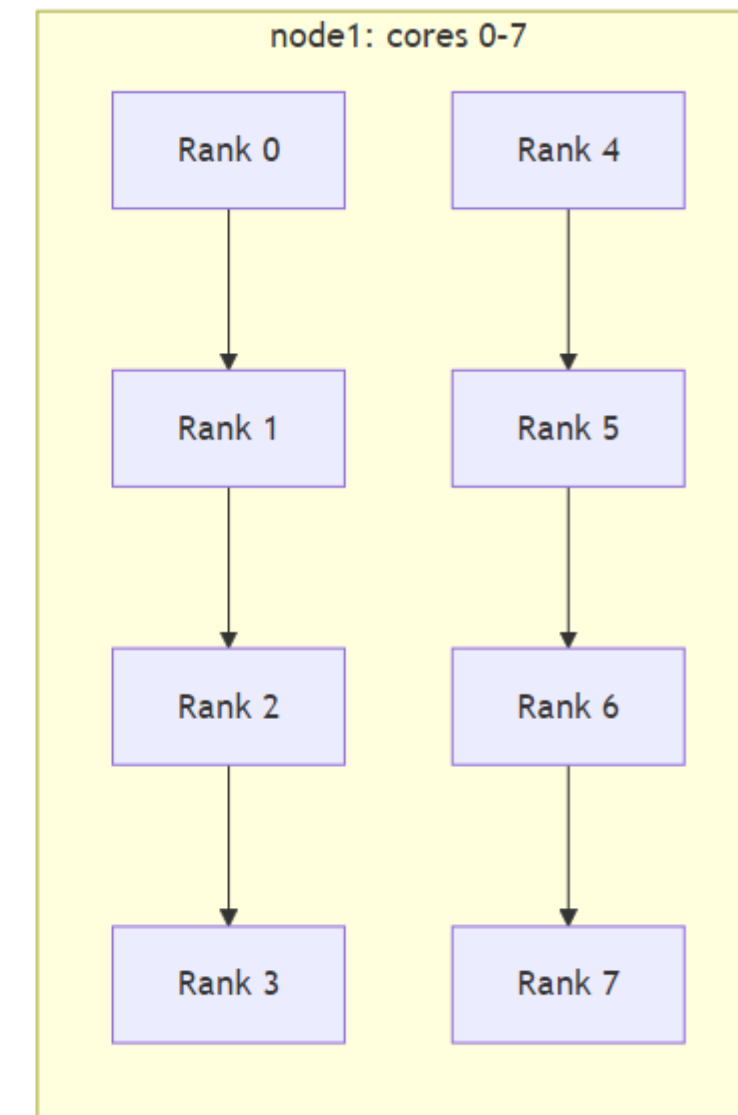
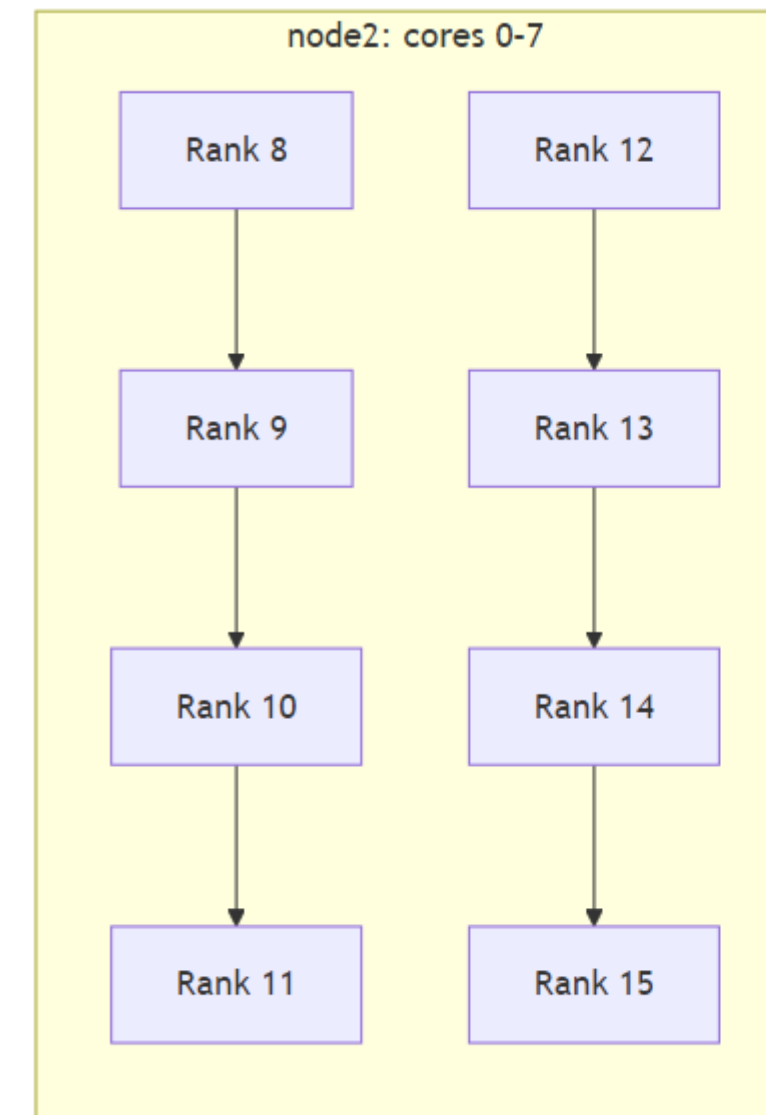
- 행렬을  $128 \times 128$  블록으로 나누고,  $(i, j)$  블록은  $(i \bmod P, j \bmod Q)$  위치의 랭크가 담당
- 각 랭크는 자신에게 할당된 블록만 계산 및 소유  $\rightarrow$  통신과 계산 균형 확보

## 4. 테스트

### MPI와 P, Q

#### 실행 과정 요약

- $1024 \times 1024$  행렬  $\rightarrow 128 \times 128$  블록으로 분할 ( $8 \times 8 = 64$ 개)
- 16 랭크가 블록을 개별 소유하며 계산
- Panel Factorization  $\rightarrow$  Broadcast (row-wise 범위)
- Row Pivoting + Update (column-wise 범위)  $\rightarrow$  DGEMM 연산 등 협업 수행
- MPI 통신을 통해 필요한 블록의 데이터를 각 랭크가 가져옴
- 최종 LU 분해 완료  $\rightarrow$  성능은  $(2/3 * 1024^3) / \text{시간으로 계산}$



## 4. 테스트

### HPL 테스트 예 2.

- $N=2048$
- $NB=128$
- $P=4, Q=4$
- node1 8개 코어 node2 8개 코어
- `export OMP_NUM_THREADS=2`

## 4. 테스트

### HPL 테스트 예 2.

- 행렬 크기:  $2048 \times 2048$
- 블록 크기:  $128 \times 128 \rightarrow$  총  $16 \times 16 = 256$  블록
- MPI 랭크 수:  $P \times Q = 4 \times 4 = 16$
- OpenMP 스레드 수: 랭크당 2 스레드  
 $\rightarrow$  총 스레드 수:  $16 \text{ 랭크} \times 2 = 32 \text{ 스레드}$
- 코어 자원: node1: 8코어  
node2: 8코어  
= 총 16 physical cores

$\rightarrow$  각 노드에 랭크 8개 VS OpenMP 2스레드 = 16 스레드  $\rightarrow$  각 노드 스레드 수 초과(over-subscribe)

## 4. 테스트

### HPL 테스트 예 2.

#### 1. 코어 오버-구독(oversubscribe) 문제

- 32 스레드 > 16 physical core 이므로 **Full oversubscription 상황**
- OS가 스레드 스케줄링을 통해 코어를 분할해 사용하게 되며,  
일반적으로 **성능 저하 발생**

## 4. 테스트

### HPL 테스트 예 2.

#### 2. MPI + OpenMP 하이브리드 설정 예시

```
#!/bin/bash
export OMP_NUM_THREADS=2
export OMP_PROC_BIND=TRUE
export OMP_PLACES=cores
```

```
mpirun -np 16 \
  --map-by ppr:8:node:PE=2 \
  --bind-to core \
  ./xhpl
```

- ppr:8:node:PE=2 → 노드당 8 랭크, 랭크당 2 스레드
- bind-to core와 OMP\_PROC\_BIND=TRUE 세팅은 핵심적이며,
- 하지만 물리 코어가 부족해 **멀티-스레드 컨텍스트 스위칭 비용 증가**

## 4. 테스트

### HPL 테스트 예 3.

- 행렬 크기:  $2048 \times 2048$
- 블록 크기:  $128 \times 128 \rightarrow$  총  $16 \times 16 = 256$  블록
- MPI 랭크 수:  $P \times Q = 4 \times 4 = 16$
- OpenMP 스레드 수: 랭크당 2 스레드  
 $\rightarrow$  총 스레드 수:  $16 \text{ 랭크} \times 2 = 32 \text{ 스레드}$
- 코어 자원: node1: 16코어,  
node2: 16코어  
 $=$  총 32 physical cores

$\rightarrow$  스레드 수 = 코어 수이므로 oversubscribe 없이 자원이 충분히 매핑됨  
스레드와 코어가 균형, 따라서 성능 저하나 스케줄링 오버헤드는 없다고 볼 수 있음

## 4. 테스트

### HPL 테스트 예 3.

#### 2. 스레드와 랭크 매핑 설정 예시

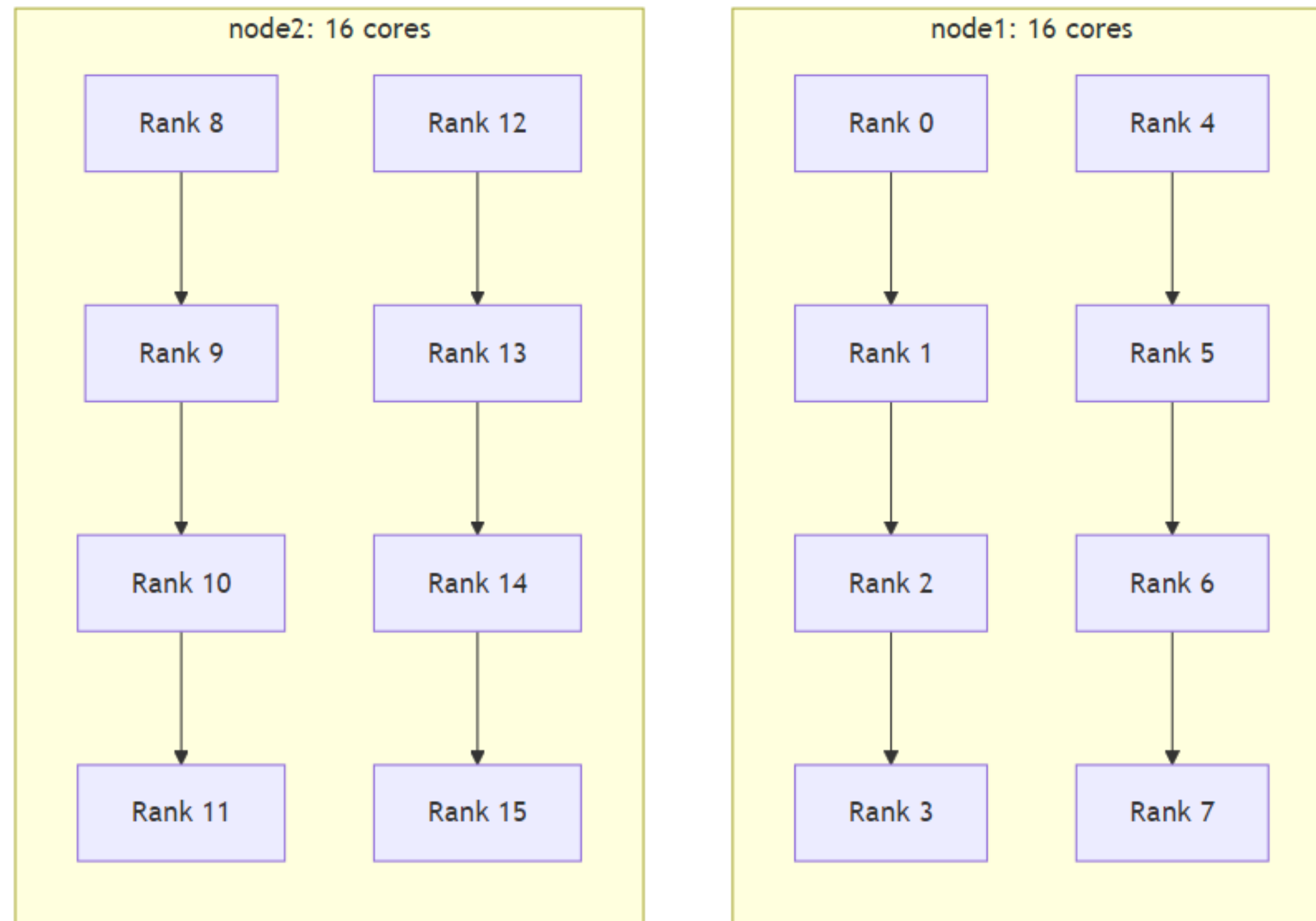
```
#!/bin/bash
export OMP_NUM_THREADS=2
export OMP_PROC_BIND=TRUE
export OMP_PLACES=cores
```

```
mpirun -np 16 \
  --map-by ppr:8:node:PE=2 \
  --bind-to core \
  ./xhpl
```

- ppr:8:node:PE=2: 노드당 8 랭크, 각 랭크에 2 코어 할당 → 총 16 스레드/랭크
- --bind-to core + OMP\_PLACES=cores 설정으로 프로세스와 스레드가 고정된 코어에서 실행됨
- 결과적으로 스레드 간 캐시 경쟁 최소화, 성능 예측성 확보

## 4. 테스트

### HPL 테스트 예 3.



## 4. 테스트

### HPL 최적화 - Hybrid MPI+OpenMP

#### 1. 성능 모델링 & 타이밍 분석

- HPL은 계산 집약적 성능 계선으로 (DGEMM 등등),  $R_{max} \approx R_{peak} \times \text{효율 수준}$ 까지 도달 가능하며, 통신 병목은 대부분 Panel Broadcast/Update에서 발생
- Hybrid 구성 시, MPI 전역 통신(collectives)는 여전히 MPI 레벨에서 병렬 처리되며, OpenMP는 DGEMM 내부에서 노드 내 병렬화 담당
- 성능 분석 결과:
  - CPU-only single node 환경에서 Hybrid MPI+OpenMP는 좋은 성능 효율( $\approx 80\%$  이상) 확보 가능
  - 반면 병렬 네트워크 환경에서는 오히려 Pure MPI가 더 우수한 성능을 보이기도 하며, OpenMP 병합은 케이스에 따라 연구가 필요

## 4. 테스트

### HPL 최적화 - Hybrid MPI+OpenMP

#### 2. mpirun + Thread Binding 최적화 팁

- MPI\_INIT\_THREAD with MPI\_THREAD\_MULTIPLE 사용: Hybrid 환경에서는 반드시 요청
- 프로세스/스레드 바인딩 전략:
  - `--map-by ppr:8:node:PE=2 + --bind-to core + OMP_PROC_BIND=TRUE, OMP_PLACES=cores` 추천
  - 또는 MVAPICH2의 `MV2_CPU_BINDING_POLICY=hybrid, MV2_THREADS_PER_PROCESS=2, MV2_THREADS_BINDING_POLICY=compact` 설정이 효과적
- 바인딩 레벨 선택:
  - `cores`: 스레드가 각각 다른 코어에 고정되어 혼잡 최소화
  - `sockets`: 소켓 단위로 묶어 NUMA 효과 감소 [UL HPC Tutorials](#).
- OpenMPI 주의 사항:
  - 기본 `--bind-to cores`는 MPI 랭크만 코어 하나에 고정 → OpenMP 스레드 모두 같은 코어에서 병목 발생
- 해결 방법: `--bind-to none` 사용 + `OMP_PROC_BIND, OMP_PLACES`로 직접 바인딩 제어.

## 4. 테스트

### HPL 최적화 - Hybrid MPI+OpenMP

구성	세팅	장점	단점/병목
Pure MPI	랭크 = 코어 수, OMP=1	MPI collectives 병렬화 우수, 단순	노드 내 멀티코어 미활용
Hybrid MPI+OpenMP	ex. 랭크=8, OMP=2	노드 내 DGEMM 가속, MPI 통신 절약	바인딩·통신 최적화 실패 시 성능 저하

## 4. 테스트

HPL

/opt/intel/oneapi/mkl/2024.1/share/mkl/benchmarks/mp\_linpac

## 4. 테스트

HPL

[https://www.advancedclustering.com/act\\_kb/tune-hpl-dat-file/](https://www.advancedclustering.com/act_kb/tune-hpl-dat-file/)

## 4. 테스트

### HPL

← → ↺ 🔒 advancedclustering.com/act\_kb/tune-hpl-dat-file/ 🏠 🗃️ 🗨️ ☆ 🌐 🎨 📄 🔄 🧩 👤 ⋮

# HOW DO I TUNE MY HPL.DAT FILE?

Tuning HPL can be a long and difficult process. Once you've found the perfect BLAS library for your architecture, now you need to create a perfect HPL.dat file. Use the form below and generate an output file as a starting point on getting the best GFLOP number you can out of your cluster.

Input

Nodes:

Cores per Node:

Memory per Node (MB):

Block Size (NB):

192

Go!

Categories

> Getting Support (5)

> Hardware (35)

> Areca Raid Arrays (3)

> InfiniBand (10)

> LSI Raid Arrays (9)

> NVIDIA Graphics Cards (1)

> Racks (1)

> Troubleshooting (8)

> Software (11)

> ACT Utilities (5)


> HPC apps & benchmarks (1)


> Linux (3)

> Schedulers (3)

> SGE / Grid Engine (1)

> TORQUE (1)

 HPC 이노베이션 허브

 한국컴퓨팅산업협회  
Korea Computing Industry Association

# 4. 테스트

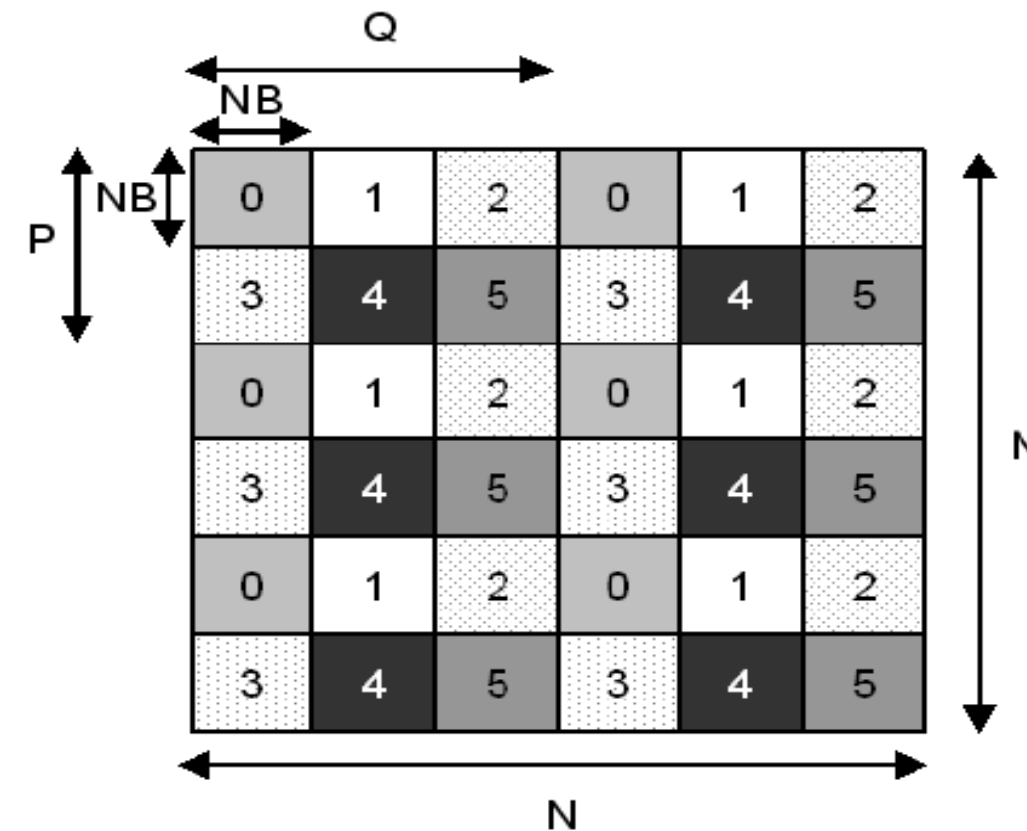
## HPL

HPLinpack benchmark input file  
Innovative Computing Laboratory, University of Tennessee  
HPL.out        output file name (if any)  
6               device out (6=stdout,7=stderr,file)  
1               # of problems sizes (N)  
**200832        Ns**  
1               # of NBs  
**192 256        NBs**  
1               PMAP process mapping (0=Row-,1=Column-major)  
1               # of process grids (P x Q)  
**1 2            Ps**  
**1 2            Qs**  
16.0            threshold  
1               # of panel fact  
2 1 0           PFACTs (0=left, 1=Crout, 2=Right)  
1               # of recursive stopping criterium  
2               NBMINs (>= 1)  
1               # of panels in recursion  
2               NDIVs  
1               # of recursive panel fact.  
1 0 2           RFACTs (0=left, 1=Crout, 2=Right)  
1               # of broadcast  
0               BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)  
1               # of lookahead depth  
0               DEPTHS (>=0)  
0               SWAP (0=bin-exch,1=long,2=mix)  
1               swapping threshold  
1               L1 in (0=transposed,1=no-transposed) form  
1               U in (0=transposed,1=no-transposed) form  
0               Equilibration (0=no,1=yes)  
8               memory alignment in double (> 0)

## 4. 테스트

### HPL

An example of P by Q partitioning of a HPL matrix in 6 processes (2x3 decomposition)  
6개의 프로세스(2x3 분해)에서 HPL 행렬의 P x Q 분할 예시(2x3 분해)



## 4. 테스트

### HPL

Blocking size (NB) recommendation for Intel(R)  
Distribution for LINPACK\* Benchmark binary

Recommended blocking sizes (NB in HPL.dat) are listed below for various Intel(R)  
architectures:

Intel(R) Xeon(R) Processor X56*/E56*/E7-*/E7*/X7*	: 256
Intel(R) Xeon(R) Processor E26*/E26* v2	: 256
Intel(R) Xeon(R) Processor E26* v3/E26* v4	: 192
Intel(R) Xeon(R) Scalable Processors	: 384
Intel(R) Xe(R) Graphics Card	: 384

Blocking size (NB) recommendation for Intel(R)  
Optimized HPL-AI\* Benchmark binary

Recommended blocking sizes (NB in HPL.dat) are listed below for various Intel(R)  
architectures:

Intel(R) Xeon(R) Processor X56*/E56*/E7-*/E7*/X7*	: 256
Intel(R) Xeon(R) Processor E26*/E26* v2	: 256
Intel(R) Xeon(R) Processor E26* v3/E26* v4	: 192
Intel(R) Xeon(R) Scalable Processors	: 384
Intel(R) 3rd Generation Xeon(R) Scalable Processors	: 768
Intel(R) 4th Generation Xeon(R) Scalable Processors	: 1536
Intel(R) Xe(R) Graphics Card	: 1152

## 4. 테스트

### HPL

시스템 성능을 최대한으로 높이려면 메모리에 가장 큰 문제 크기를 맞추는 것이 목표

HPL은  $N \times N$  배열의 이중 정밀도(DP) 소자를 계산하고 각 이중 정밀도 소자는 (DP) = 8바이트의 크기를 요구하기 때문에  $N$ 의 문제 크기에 소비되는 메모리는  $8N^2$ 이다.

$$\text{sqrt}((\text{Memory Size in Gbytes} * 1024 * 1024 * 1024 * \text{Number of Nodes}) / 8) * 0.92$$

## 4. 테스트

HPL

```
watch -n.1 "grep \"^[c]pu MHz\" /proc/cpuinfo"
```

## 4. 테스트

### HPL

```
mpirun -ppn 1 -np 1 ./xhpl_intel64_dynamic
```

## 4. 테스트

### HPL

Architecture: x86\_64  
CPU op-mode(s): 32-bit, 64-bit  
Byte Order: Little Endian  
CPU(s): 96  
On-line CPU(s) list: 0-95  
Thread(s) per core: 1  
Core(s) per socket: 48  
Socket(s): 2  
NUMA node(s): 2  
Vendor ID: GenuineIntel  
BIOS Vendor ID: Intel(R) Corporation  
CPU family: 6  
Model: 143  
Model name: Intel(R) Xeon(R) Platinum 8468  
BIOS Model name: Intel(R) Xeon(R) Platinum 8468  
Stepping: 8  
CPU MHz: 2100.000  
CPU max MHz: 3800.0000  
CPU min MHz: 800.0000  
BogoMIPS: 4200.00  
L1d cache: 48K  
L1i cache: 32K  
L2 cache: 2048K  
L3 cache: 107520K  
NUMA node0 CPU(s): 0-47  
NUMA node1 CPU(s): 48-95

## 4. 테스트

### HPL

```
=====
T/V          N  NB  P  Q          Time          Gflops
-----
WC00C2R2     150000 384  1  1          399.62          5.63049e+03
HPL_pdgesv() start time Tue Jun 11 00:20:47 2024

HPL_pdgesv() end time  Tue Jun 11 00:27:26 2024

-----
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)= 3.34072432e-03 ..... PASSED
=====
```

Finished 1 tests with the following results:  
1 tests completed and passed residual checks,  
0 tests completed and failed residual checks,  
0 tests skipped because of illegal input values.

$$2.1 * 96 * 32 = 6,451$$

## 4. 테스트

### HPL

```
=====
T/V          N  NB  P  Q          Time          Gflops
-----
WC00C2R2    150000 384  1  2          216.89          1.03738e+04
```

HPL\_pdgesv() start time Tue Jun 11 00:38:26 2024

HPL\_pdgesv() end time Tue Jun 11 00:42:03 2024

```
-----
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)= 3.19009359e-03 ..... PASSED
=====
```

Finished 1 tests with the following results:

1 tests completed and passed residual checks,  
0 tests completed and failed residual checks,  
0 tests skipped because of illegal input values.

$$2.1 * 96 * 32 * 2 = 12,902$$

# 4. 테스트

## HPL – BIOS

Configuration Item	Recommended Value
Hyper-Threading (SMT)	Enabled* (see text)
Core Prefetchers	Enabled*
Turbo Boost Technology	Enabled*
Intel® SpeedStep® (P-States)	Disabled
SNC (Sub-NUMA Clusters)	Enabled
IMC Interleave	One-way
UPI Prefetch	Enabled*
XPT Prefetch	Enabled*
Total Memory Encryption (TME)	Disabled
Memory controller page policy	Static closed
Autonomous Core C-State	Disabled*
CPU C6 Report	Disabled*
Enhanced Halt State (C1E)	Disabled*
Package C State	C0/C1 State*
Relax Ordering	Disabled*
Intel VT for Directed I/O (Intel VT-D)	Disabled*
CPU Power Policy	Performance
Local/Remote Threshold	Auto*
LLC Prefetch	Disabled*

Configuration Item	Recommended Value
LLC Dead Line Alloc.	Enabled*
Directory AToS	Disabled*
Direct-to-UPI (D2U)	Enabled
DBP-for-F	Enabled

# 4. 테스트

## HPL – BIOS (Dell PowerEdge Servers)

System setup screen	Setting	Default	Recommended setting for performance for HPC and SPECcpu speed environments		Recommended setting for low latency, Stream, and MLC environments	Recommended for general business/scientific throughput (for example, SPECcpu2017)
System profile settings	System Profile	Performance Per Watt [1]	Performance Optimized		First select Performance Optimized and then select Custom [1]	Custom
System profile settings	CPU Power Management	System DBPM	Maximum Performance		Maximum Performance	Maximum Performance
System profile settings	Memory Frequency	Maximum Performance	Maximum Performance		Maximum Performance	Maximum Performance
System profile settings	Turbo Boost [2]	Enabled	Enabled		Enabled	Enabled
System profile settings	C1E	Enabled	Disabled		Disabled	Disabled
System profile settings	C States	Enabled	Disabled		Disabled	Autonomous or Disabled [6]
System profile settings	Monitor/Mwait	Enabled	Enabled		Disabled [3]	Enabled
System profile settings	Memory Patrol Scrub	Standard	Standard [4]		Standard/Disabled [4]	Disabled
System profile settings	Memory Refresh Rate	1x	1x		1x	1x
System profile settings	Uncore Frequency	Dynamic	Maximum [5]		Maximum [5]	Dynamic
System profile settings	Energy Efficient Policy	Balanced Performance	Performance		Performance	Performance
System profile settings	CPU Interconnect Bus Link Power Management	Enabled	Disabled		Disabled	Disabled
System profile settings	PCI ASPM L1 Link Power Management	Enabled	Disabled		Disabled	Disabled

## 4. 테스트

### HPL – BIOS (Dell PowerEdge Servers)

System setup screen	Setting	Default	Recommended setting for performance for HPC and SPECcpu speed environments	Recommended setting for low latency, Stream, and MLC environments	Recommended for general business/scientific throughput (for example, SPECcpu2017)
Memory settings	Memory Operating Mode	Optimizer	Optimizer [1]	Optimizer [1]	Optimizer [1]
Memory settings	Memory Node Interleave	Disabled	Disabled	Disabled	Disabled
Memory settings	DIMM Self Healing	Enabled	Disabled	Disabled	Disabled
Memory settings	ADDDC setting	Disabled [2]	Disabled [2]	Disabled [2]	Disabled [2]
Memory settings	Memory Training	Fast	Fast	Fast	Fast
Memory settings	Correctable Error Logging	Enabled	Disabled	Disabled	Disabled
Processor settings	Logical Processor	Enabled	Disabled [3]	Disabled [3]	Enabled
Processor settings	Virtualization Technology	Enabled	Disabled	Disabled	Disabled
Processor settings	CPU Interconnect Speed	Maximum Data Rate	Maximum Data Rate	Maximum Data Rate	Maximum Data Rate
Processor settings	Adjacent Cache Line Prefetch	Enabled	Enabled	Enabled	Enabled
Processor settings	Hardware Prefetcher	Enabled	Enabled	Enabled	Enabled
Processor settings	DCU Streamer Prefetcher	Enabled	Enabled	Disabled	Disabled
Processor settings	DCU IP Prefetcher	Enabled	Enabled	Enabled	Enabled
Processor settings	Sub NUMA Cluster	Disabled	SNC 2	SNC 4 on XCC SNC 2 on MCC	SNC 4 on XCC SNC 2 on MCC
Processor settings	Dell Controlled Turbo	Disabled	Disabled	Enabled [4]	Disabled
Processor settings	Dell Controlled Turbo Optimizer mode	Disabled	Enabled [5]	Enabled [5]	Enabled [5]
Processor settings	XPT Prefetch	Enabled	Disabled	Disabled	Enabled
Processor settings	UPI Prefetch	Enabled	Disabled	Disabled	Enabled
Processor settings	LLC Prefetch	Disabled	Enabled	Disabled	Disabled
Processor settings	DeadLine LLC Alloc	Enabled	Enabled	Enabled	Disabled
Processor settings	Directory AtoS	Disabled	Disabled	Disabled	Disabled
Processor settings	Dynamic SST Perf Profile	Disabled	Disabled	Enabled	Disabled
Processor settings	SST-Perf- profile	Operating Point 1	Operating Point 1	Operating Point ? [6]	Operating Point 1
iDRAC settings	Thermal Profile	Default	Maximum Performance	Maximum Performance	Maximum Performanc

## 4. 테스트

<https://sp.ts.fujitsu.com/dmsp/Publications/public/wp-performance-report-primergy-rx2540-m6->

### HPL – BIOS 2 (Fujitsu Server PRIMERGY RX2540 M6.))

System Under Test (SUT)	
<b>Hardware</b>	
• Model	PRIMERGY RX2540 M6
• Processor	2 x 3rd Generation Intel Xeon Scalable Processors Family
• Memory	32 x 32 GB 2Rx4 PC4-3200AA-R
<b>Software</b>	
• BIOS settings	<ul style="list-style-type: none"> <li>• HyperThreading = Disabled</li> <li>• Link Frequency Select = 10.4 GT/s</li> <li>• HWPM Support = Disabled</li> <li>• Intel Virtualization Technology = Disabled</li> <li>• LLC Dead Line Alloc = Disabled</li> <li>• Stale AtoS = Enabled</li> <li>• Fan Control = Full</li> </ul>
• Operating system	Red Hat Enterprise Linux Server release 8.2 4.18.0-193.el8.x86_64
• Operating system settings	Kernel Boot Parameter set with : nohz_full=1-X (X: logical core number -1) cpupower -c all frequency-set -g performance echo 50000 > /proc/sys/kernel/sched_cfs_bandwidth_slice_us echo 240000000 > /proc/sys/kernel/sched_latency_ns echo 5000000 > /proc/sys/kernel/sched_migration_cost_ns echo 100000000 > /proc/sys/kernel/sched_min_granularity_ns echo 150000000 > /proc/sys/kernel/sched_wakeup_granularity_ns echo always > /sys/kernel/mm/transparent_hugepage/enabled echo 1048576 > /proc/sys/fs/aio-max-nr run with avx512
• Compiler	C/C++: Version 19.1.2.254 of Intel C/C++ Compiler for Linux
• Benchmark	Intel Optimized MP LINPACK Benchmark for Clusters

Processor	Number of cores	Rated frequency [GHz]	Number of processors	Rpeak [GFlops]	Rmax [GFlops]	Efficiency
Xeon Platinum 8380	40	2.3	2	5,888	4,431	75%
Xeon Platinum 8368Q	38	2.6	2	6,323	4,406	70%
Xeon Platinum 8368	38	2.4	2	5,837	4,249	73%
Xeon Platinum 8362 <sup>*1</sup>	32	2.8	2	5,734	4,040	70%
Xeon Platinum 8360Y	36	2.4	2	5,530	3,938	71%
Xeon Platinum 8358P	32	2.6	2	5,325	3,560	67%
Xeon Platinum 8358	32	2.6	2	5,325	3,786	71%
Xeon Platinum 8352Y	32	2.2	2	4,506	3,166	70%
Xeon Platinum 8352V	36	2.1	2	4,838	3,327	69%
Xeon Platinum 8352M <sup>*1</sup>	32	2.3	2	4,710	2,939	62%
Xeon Gold 6354	18	3.0	2	3,456	2,446	71%
Xeon Gold 6348	28	2.6	2	4,659	3,350	72%
Xeon Gold 6346	16	3.1	2	3,174	2,326	73%
Xeon Gold 6342 <sup>*1</sup>	24	2.8	2	4,301	3,173	74%
Xeon Gold 6338T <sup>*1</sup>	24	2.1	2	3,226	2,363	73%
Xeon Gold 6338	32	2.0	2	4,096	3,156	77%
Xeon Gold 6336Y <sup>*1</sup>	24	2.4	2	3,686	2,657	72%
Xeon Gold 6334 <sup>*1</sup>	8	3.6	2	1,843	1,316	71%
Xeon Gold 6330N	28	2.2	2	3,942	2,506	64%
Xeon Gold 6330	28	2.0	2	3,584	2,994	84%
Xeon Gold 6326 <sup>*1</sup>	16	2.9	2	2,970	2,204	74%
Xeon Gold 6314U	32	2.3	1	2,355	1,592	68%
Xeon Gold 6312U <sup>*1</sup>	24	2.4	1	1,843	1,341	73%
Xeon Gold 5320 <sup>*1</sup>	26	2.2	2	3,661	2,816	77%
Xeon Gold 5318Y <sup>*1</sup>	24	2.1	2	3,226	2,456	76%
Xeon Gold 5318S <sup>*1</sup>	24	2.1	2	3,226	2,457	76%
Xeon Gold 5317 <sup>*1</sup>	12	3.0	2	2,304	1,614	70%
Xeon Gold 5315Y <sup>*1</sup>	8	3.2	2	1,638	1,211	74%
Xeon Silver 4316 <sup>*1</sup>	20	2.3	2	2,944	2,085	71%
Xeon Silver 4314 <sup>*1</sup>	16	2.4	2	2,458	1,702	69%
Xeon Silver 4310 <sup>*1</sup>	12	2.1	2	1,613	1,480	92%
Xeon Silver 4309Y <sup>*1</sup>	8	2.8	2	1,434	1,026	72%

\*1: To be supported in July 2021 or later