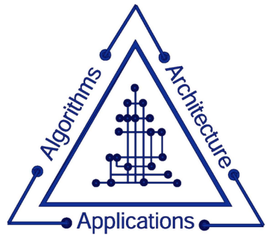
BETTER TOOLS FOR BETTER P^3

Michael Wolfe

NVIDIA Corp.

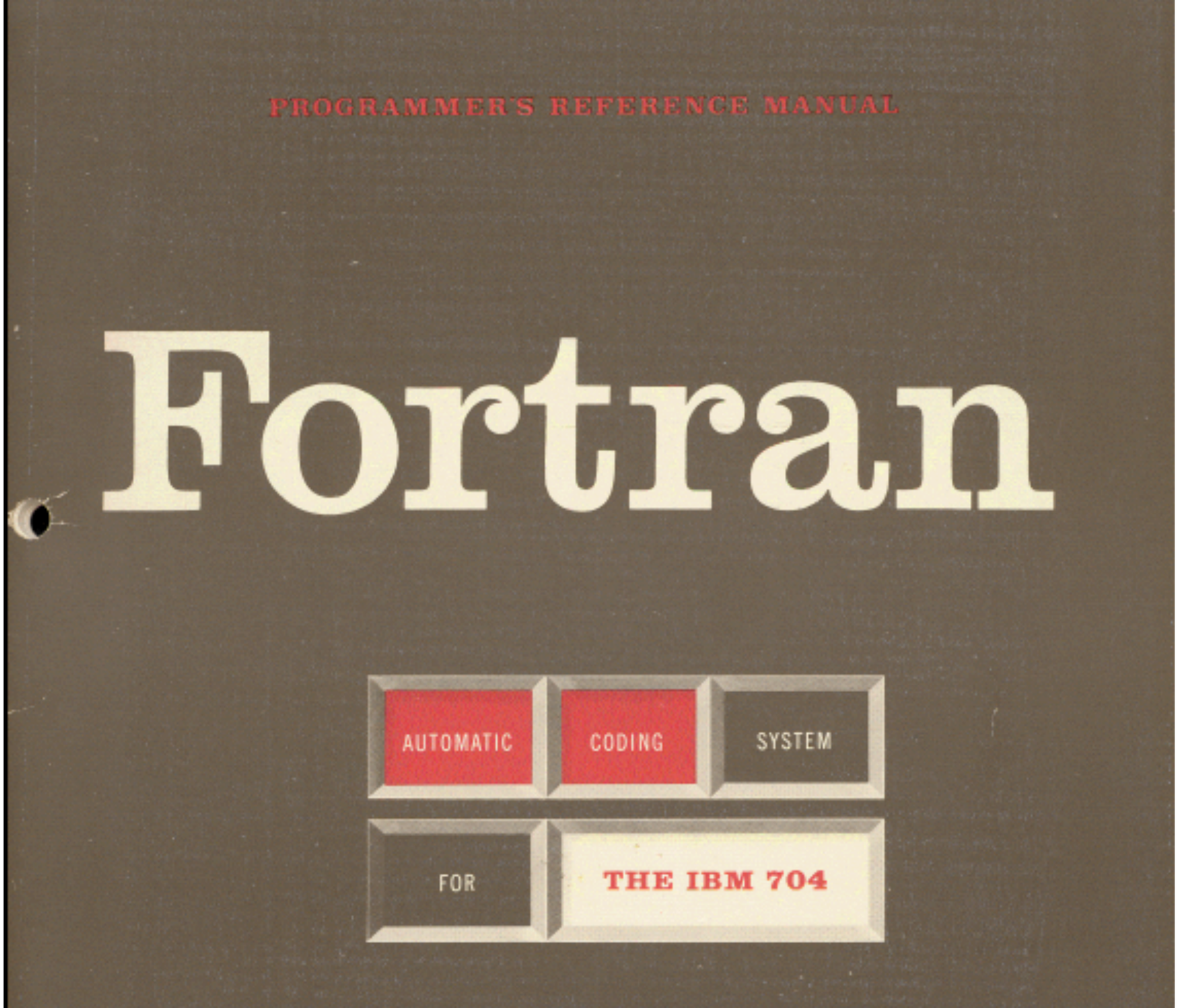


40TH ANNIVERSARY LECTURE SERIES

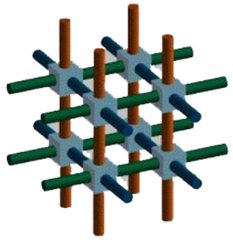
Programming at Exascale

Michael Wolfe
The Portland Group, Inc.

April 2011

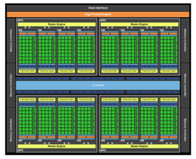

PROGRAMMER'S REFERENCE MANUAL
Fortran
AUTOMATIC
CODING
SYSTEM
FOR
THE IBM 704

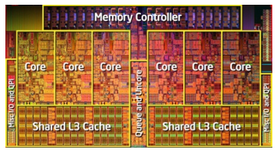
SIX LEVELS OF PARALLELISM



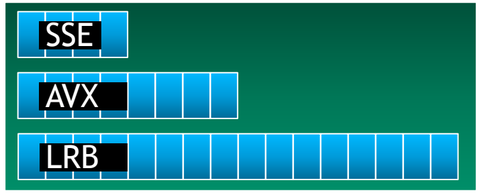
Node

Socket





Core



Vector

IF	ID	RF	EX	EX	EX	WB

Pipeline

IF	ID	RF	EX	EX	EX	WB	
IF	ID	RF	EX	EX	WB		
IF	ID	RF	EX	EX	EX	EX	WB

Instruction

CRAY

RESEARCH INC.

NOTE: this tech note
is obsolete ...
but can be xeroxed
for reference
material. . .

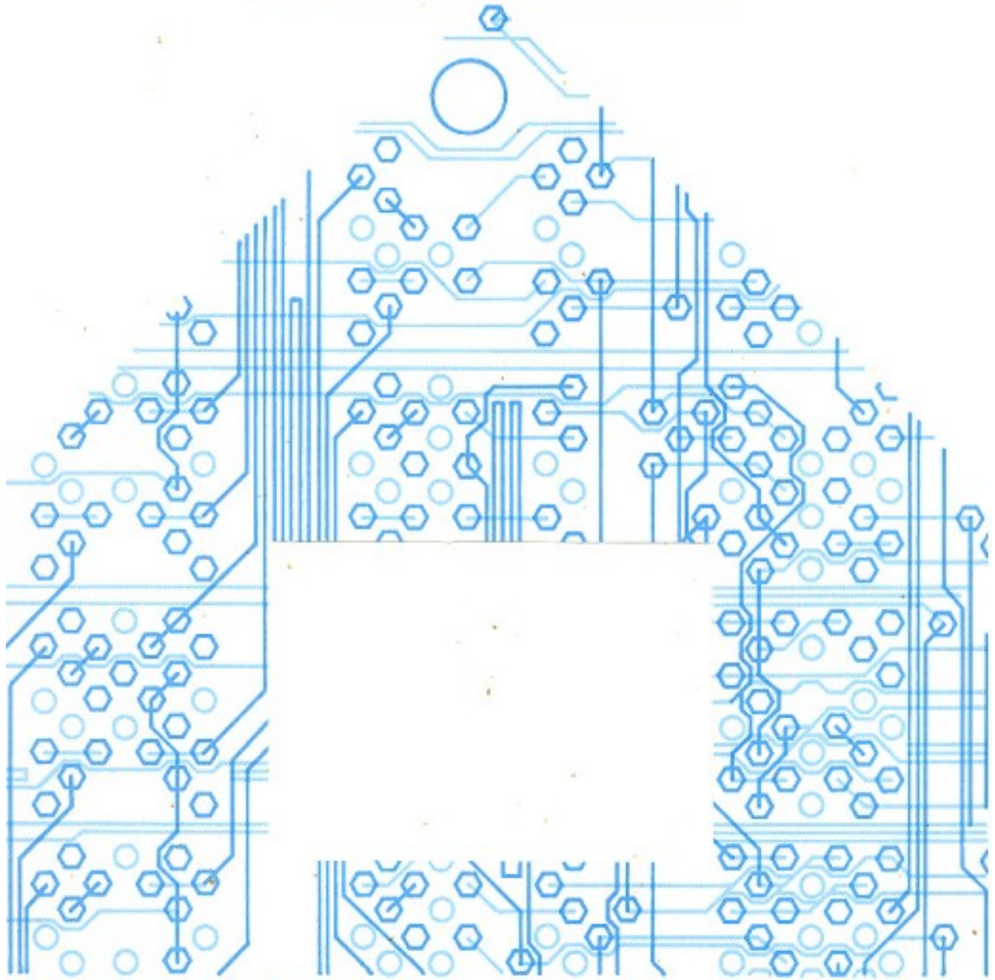
Vectorization and Conversion of Fortran Programs for the CRAY-1 (CFT) Compiler

by

Lee Higbie

2240207

CRAY RESEARCH, INC.



TECHNICAL NOTE

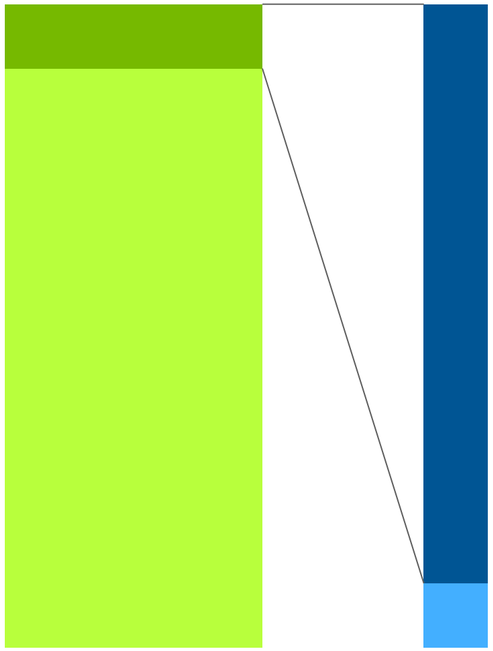
```
do j = lo, hi

. . . . .

enddo
```

```
subroutine bum(...)
 do i = ...
  do j = lo, hi
   do k = ...
    call ble(.....)
   enddo
  enddo
 enddo
end subroutine
...
do m = ...
 call bum(...)
enddo
```

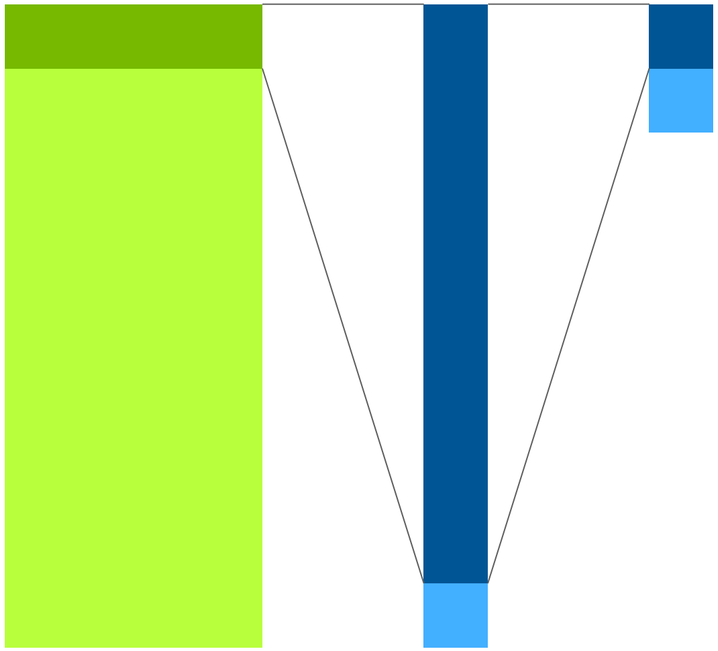
```
subroutine bum(...)
 do i = ...
  do j = lo, hi
   do k = ...
    call ble(.....)
   enddo
  enddo
 enddo
end subroutine
...
do m = ...
 call bum(...)
enddo
```

Expose
Express
Exploit

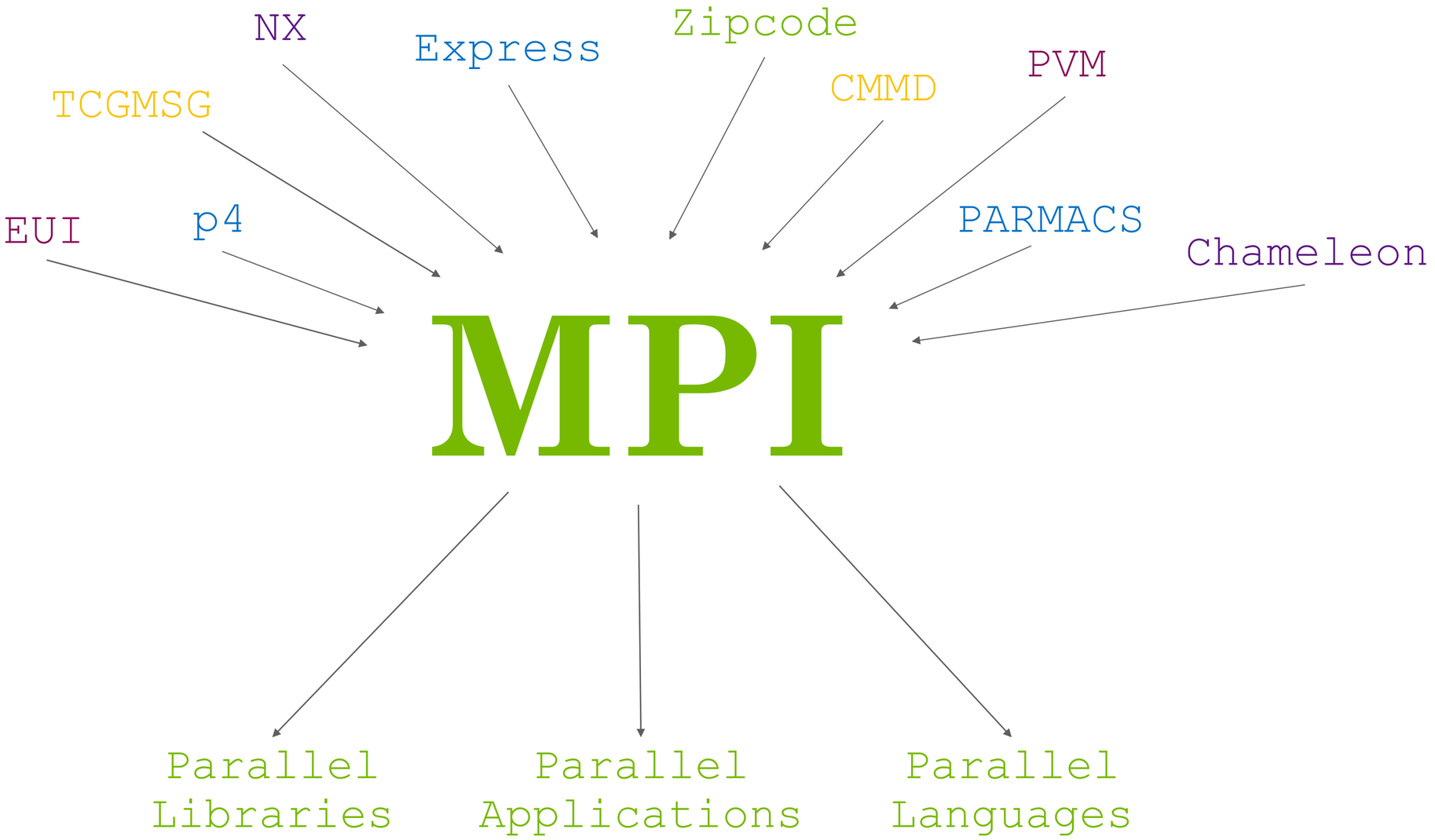
90-10 RULE AND AMDAHL'S LAW

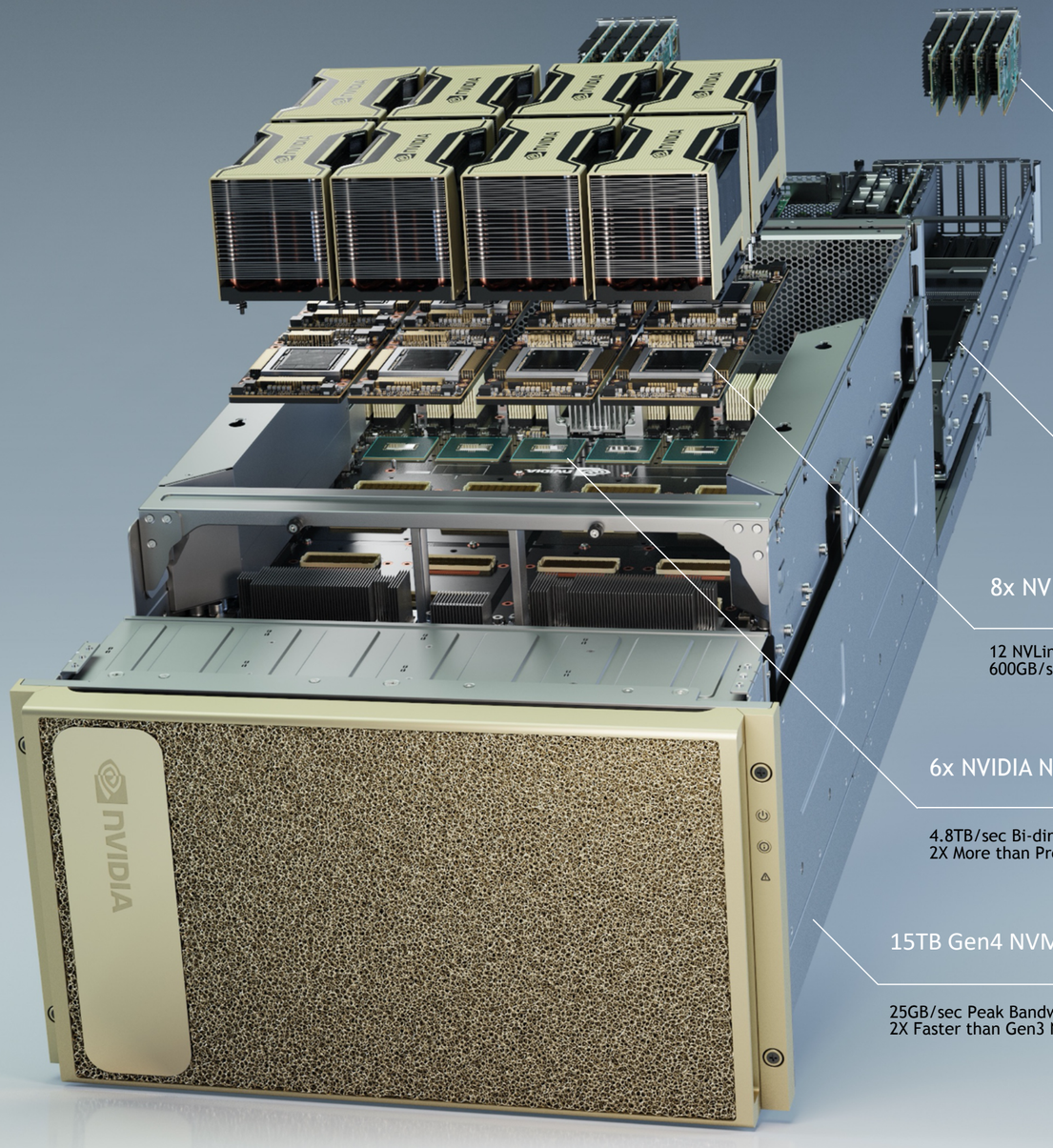


90-10 RULE AND AMDAHL'S LAW



THE MESSAGE PASSING INTERFACE STANDARD





9x Mellanox ConnectX-6 200Gb/s Network Interface

450GB/sec Peak Bi-directional Bandwidth

Dual 64-core AMD Rome CPUs and 1TB RAM

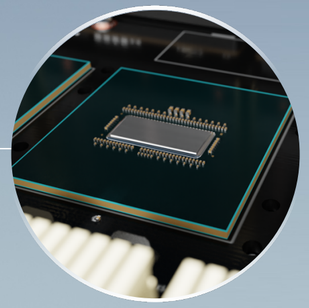
3.2X More Cores to Power the Most Intensive AI Jobs

8x NVIDIA A100 GPUs with 320GB Total GPU Memory

12 NVLinks/GPU
600GB/sec GPU-to-GPU Bi-directional Bandwidth

6x NVIDIA NVSwitches

4.8TB/sec Bi-directional Bandwidth
2X More than Previous Generation NVSwitch



15TB Gen4 NVME SSD

25GB/sec Peak Bandwidth
2X Faster than Gen3 NVME SSDs

MACHINE MODEL

Coarse-grain node-level parallelism (MPI, tasking, exposed by programmer)

Fine-grain on-node thread-level parallelism (identified by programmer)

SIMD / vector parallelism (similarly identified by programmer)

Data layout

THE FUTURE OF PARALLEL PROGRAMMING

Standard Languages

```
std::for_each_n(POL, idx(0), n,
                [&](Index_t i){
    y[i] += a*x[i];
});
```

```
do concurrent (i = 1:n)
   y(i) = y(i) + a*x(i)
enddo
```

Drive Base Languages to Better Support Parallelism

THE FUTURE OF PARALLEL PROGRAMMING

Standard Languages | Directives | Specialized Languages

```
std::for_each_n(POL, idx(0), n,
                [&](Index_t i){
    y[i] += a*x[i];
});
```

```
do concurrent (i = 1:n)
   y(i) = y(i) + a*x(i)
enddo
```

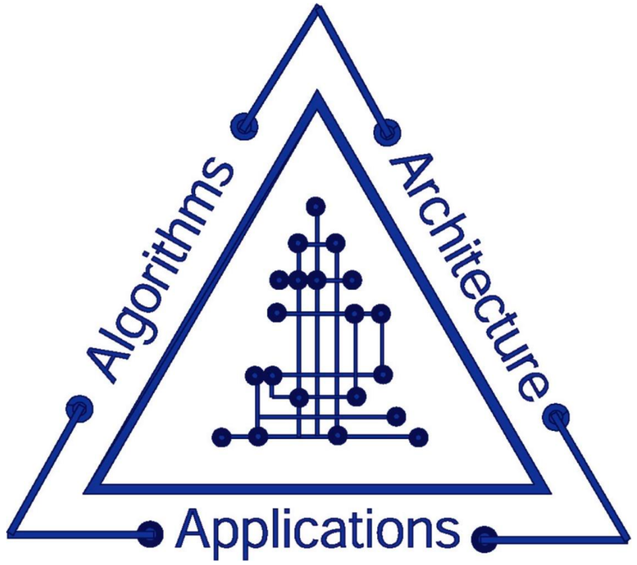
Drive Base Languages to Better Support Parallelism

```
!$acc data copy(x,y)

...

do concurrent (i = 1:n)
   y(i) = y(i) + a*x(i)
enddo

...

!$acc end data
```

Augment Base Languages with Directives When Necessary

```
attributes(global) &
subroutine saxpy(n, a, x, y)
  real :: a, x(*), y(*)
  int i = threadIdx%x + &
      (blockIdx%x-1)*blockDim%x
  if (i <= n) y(i) = y(i) + a*x(i)
end subroutine
...
  real         :: x(:), y(:)
  real,device :: d_x(:), d_y(:)
  d_x = x
  d_y = y

  call saxpy<<<(N+255)/256,256>>>&
          (n, a, d_x, d_y)
  y = d_y
```

Maximize Performance with Specialized Languages & Intrinsics



40TH ANNIVERSARY LECTURE SERIES