BETTER TOOLS FOR BETTER P³

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

by

Lee Higbie

2240207

NOTE: this tech note is obsolete but can be accessed for reference material...
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
90-10 RULE AND AMDAHL'S LAW
90-10 RULE AND AMDAHL'S LAW
THE MESSAGE PASSING INTERFACE STANDARD

MPI

TCGMSG
EUI
p4
Express
Zipcode
CMMD
PVM
PARMACS
Chameleon

Parallel Libraries
Parallel Applications
Parallel Languages
9x Mellanox ConnectX-6 200Gb/s Network Interface
450GB/sec Peak Bi-directional Bandwidth

Dual 64-core AMD Rome CPUs and 1TB RAM
3X 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

2X More than Previous Generation NVSwitch
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

\[
\text{std::for_each_n(POL, idx(0), n,}
\begin{align*}
&\quad \& (\text{Index_t } i)\{ \\
&\quad \quad y[i] += a*x[i]; \\
&\quad \}\};
\end{align*}
\]

\[
\text{do concurrent (i = 1:n)}
\begin{align*}
&\quad y(i) = y(i) + a*x(i) \\
&\text{enddo}
\end{align*}
\]

Drive Base Languages to Better Support Parallelism
THE FUTURE OF PARALLEL PROGRAMMING

Standard Languages | Directives | Specialized Languages

Drive Base Languages to Better Support Parallelism

Augment Base Languages with Directives When Necessary

Maximize Performance with Specialized Languages & Intrinsics

```c
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

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
```

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

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

```c
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
```

```c
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
```
40TH ANNIVERSARY LECTURE SERIES