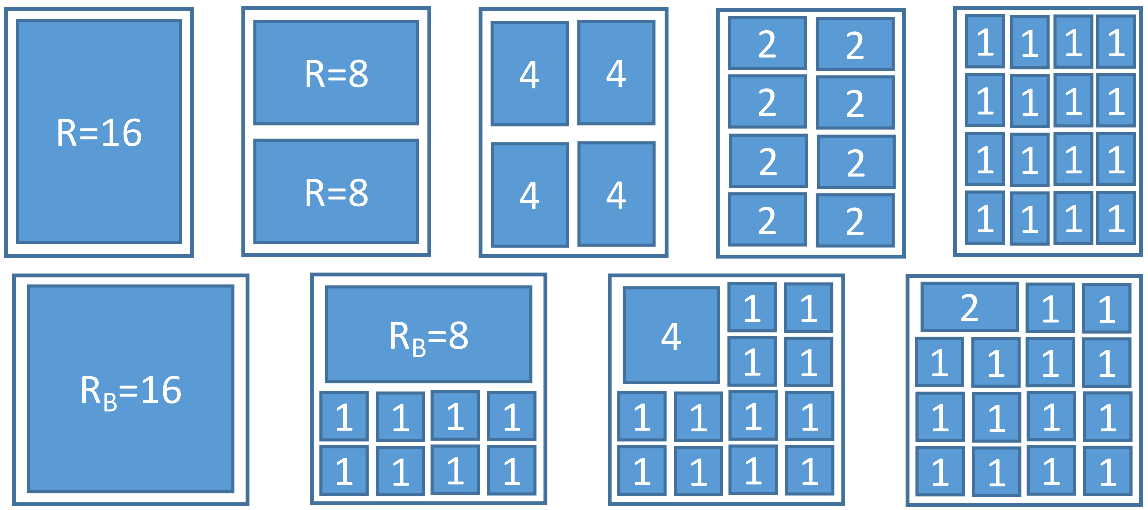
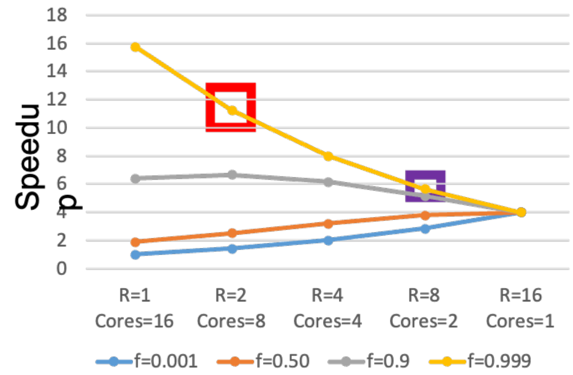
# Lecture 22: Resilience in the Exascale Era

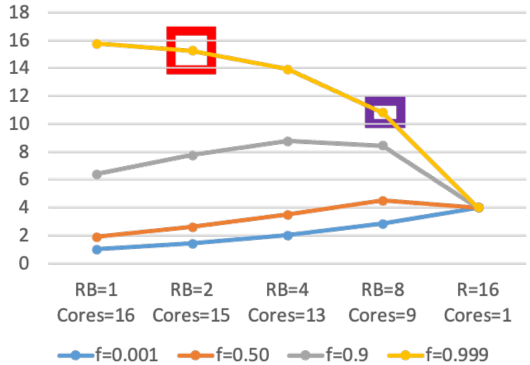
Vivek Kumar
Computer Science and Engineering
IIIT Delhi
vivekk@iiitd.ac.in

# Last Lecture (Recap)




- Symmetric multicores
  - $\text{Speedup}(f, R, N) = 1 / ( \{(1-f)/\text{Perf}_R\} + \{f / (\text{Perf}_R*(N / R))\} )$
- Asymmetric multicore
  - $\text{Speedup}(f, R_B, N) = 1 / ( \{(1-f) / \text{Perf}(R_B)\} + \{f / ( \text{Perf}(R_B) + N - R_B ) \} )$

# Today's Class

- Exascale computing
- Approaches for resilience
- Runtime solutions for resilient task-parallel programs
- Quiz-5 (Last one!)

# Exascale Computing

| Rank | System | Cores | (PFlop/s) | (PFlop/s) | (kW) |
|---|---|---|---|---|---|
| 1 | **El Capitan** - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States | 11,039,616 | 1,742.00 | 2,746.38 | 29,581 |
| 2 | **Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE DOE/SC/Oak Ridge National Laboratory United States | 9,066,176 | 1,353.00 | 2,055.72 | 24,607 |
| 3 | **Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States | 9,264,128 | 1,012.00 | 1,980.01 | 38,698 |

First exascale supercomputer (June 2022)

"APU" – CPU and GPU share the memory / LLC. Total 4 APUs/node with each APU having 24 CPU cores and one GPU

Each node with 64 CPU cores and 4 GPUs

Each node with two CPU sockets and six Intel GPUs. Total 40 Cores / Socket

# Key Challenges for Exascale

- Parallelism
  - Covered in depth in first half of this course
  - **Goal**: Support applications solving science problems 50× faster or more complex than today's 20 PF systems
- Memory and Storage
  - We covered NUMA and locality in context of the memory, but we are not covering storage in this course
  - **Goal**: Reduce memory access latency and support locality over deep memory hierarchies
- Energy Consumption
  - Covered in lectures 18 and 21
  - **Goal**: Operate in a power envelope of 20–30 MW
- Reliability
  - **Goal**: Be sufficiently **resilient** (average fault rate no worse than weekly)

# Today's Class

- Exascale computing
- Approaches for resilience
- Runtime solutions for resilient task-parallel programs

# Resilience

- It is the technique for keeping applications running to a correct solution in a timely and efficient manner despite underlying system faults
    - Exascale systems are 1,000 times more powerful will have at least 1,000 times more components and will fail 1,000 times more frequently

# Approaches for Resilience (1/5)

- Checkpointing
  - Rollback recovery approach using checkpoint/restart
  - The programmer adds some specific functions in the application to save essential state and restore from this state in case of failure
  - **Drawback:**
    - Amount of data/space needed for saving intermediate state

# Approaches for Resilience (2/5)

- Forward recovery
  - In some cases, the application can handle the error and execute some actions to terminate cleanly or follow some specific recovery procedure without relying on classic rollback recovery
    - Any example?
      - ```
        do {
            try {

            } catch (IOException e) {
                /* Take action to overcome the error and continue */
            }
        } while(condition not met);
        ```
  - **Drawback**: A prerequisite for rollforward recovery is that some application processes and the runtime environment stay alive
    - In the above example, the JVM could stay alive to complete the checkpointing upon faliure
        ```

# Approaches for Resilience (3/5)

- Replication
  - Each process is replicated such that the probability that all replicas would fail is acceptably small
  - Replicas of a process are assigned to different computers
  - They proceed asynchronously with the same code and data such that they can be viewed as an integrated logical entity by others
  - **Drawback**
    - Amount of computational resources is a major challenge
      - Usually double the number of resources actually required by the program

# Approaches for Resilience (4/5)

- Failure prediction
  - Draw conclusions about upcoming failures from the occurrence of previous failures
    - Measure the system behaviour using hardware metrics, and compare it to the expected normal behaviour using some machine learning algorithms
      - Some errors can be predicted by their side-effects on the system such as exceptional memory usage, CPU load, disk I/O, or unusual function calls in the system
      - Periodically measure such system in order to identify an imminent failure
    - **Drawback**
      - Requires some offline training runs

# Approaches for Resilience (5/5)

- Mitigating Silent Data Corruption (SDC) or Soft Errors
    - Industry-wide hardware issue impacting computer CPUs
    - An SDC occurs when an impacted CPU inadvertently causes errors in the data it processes
        - For example, an impacted CPU might miscalculate data (i.e., 1+1=3) due to manufacturing defects
            - The transistors are so tiny that small electrical fluctuations can cause errors

# Today's Class

- Exascale computing
- Approaches for resilience
- Runtime solutions for resilient task-parallel programs

# Resilience inside Task-based Runtimes

- **Use futures and promises**
  - Whenever future object gets satisfied (written by an async) then it can be checkpointed
  - For mitigating SDC, spawn an async multiple times and verify if the resultant future object from each async has exact same result
    - Programmer needs to provide an error checking function so that the runtime can use it to check for errors
  - Trace and replay

```
int i=0;
if (checkpoint exists) {
    load_checkpoint();
    i = get_iteration();
}
for(; i<MAX; i++) {
    quill::start_tracing();
    // future objects
    computation_kernel_using_async_finish();
    quill::stop_tracing();
    // Save trace data and resultant future objects
    quill::checkpoint();
}
```

# Reference Materials

- Towards exascale resilience
  - http://snir.cs.illinois.edu/listed/J53.pdf
- Enabling resilience in asynchronous many-task programming models
  - https://www.osti.gov/servlets/purl/1641008
- Exascale vision of India
  - https://amritmahotsav.negd.in/presentation/day5/Exa-scale%20Vision%20of%20India.pdf
- Silent data corruptions at scale
  - https://arxiv.org/pdf/2102.11245.pdf
- Support for Resiliency in HClib
  - https://www.osti.gov/servlets/purl/1641008

# Next Lecture

- End semester review