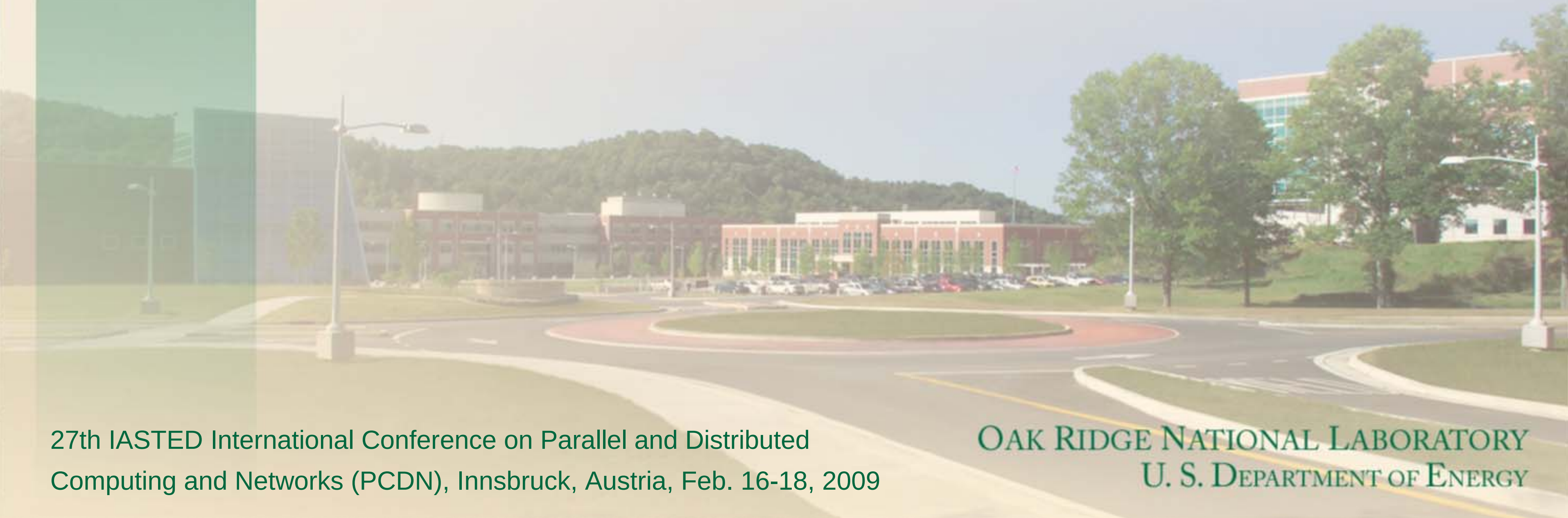
The Case for Modular Redundancy in Large-Scale High Performance Computing Systems

Christian Engelmann, Hong Ong and Stephen L. Scott

Computer Science and Mathematics Division
Oak Ridge National Laboratory



27th IASTED International Conference on Parallel and Distributed Computing and Networks (PCDN), Innsbruck, Austria, Feb. 16-18, 2009

OAK RIDGE NATIONAL LABORATORY
U. S. DEPARTMENT OF ENERGY

Trends in HPC System Reliability

- **HPC systems continue to increase in size**
  - **Error rate increases due to higher component count**
- **HPC systems may increasingly contain accelerators**
  - **Soft error rate increases due to higher vulnerability**
- **Nanometer technology continues to decrease**
  - **Soft error rate increases further due to higher vulnerability**
- **HPC vendors continue to use mass-market components**
  - **Mass-market demands define HPC system reliability**

➲ ***Future HPC systems won’t be as reliable as today’s***

➲ ***Soft errors are a major concern for HPC resilience***

Motivation for Modular Redundancy in HPC

- **Redundancy on compute nodes is not entirely new**
  - **Diskless checkpointing (Plank et al.)**
  - **Algorithmic redundancy approaches (Dongarra et al.)**
- **Until now, the HPC community (researchers and vendors) stayed away from modular redundancy**
  - **"Big hammer" approach with fully redundant compute nodes**

➲ ***With increasing hard and (especially) soft error rates, compute-node redundancy needs to be considered as an alternative to checkpointing and preemptive migration***

➲ ***Respective research and development in modular redundancy for HPC environments is needed***

Trends in HPC System Resilience

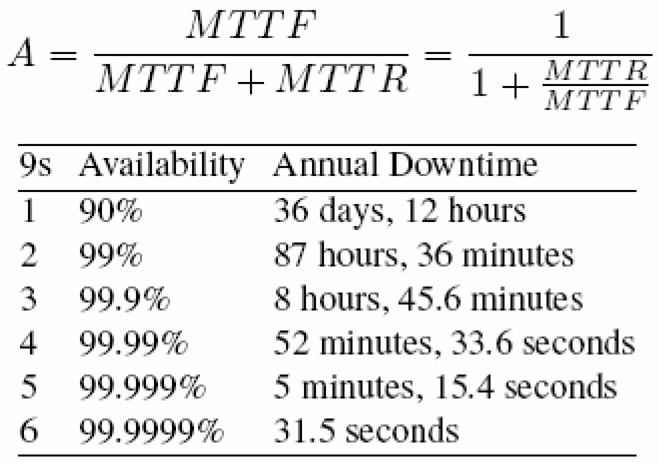
- **Checkpoint/restart has limits**
  - **Efficiency decreases with higher error rate**
  - **Efficiency decreases further with larger aggregated memory**
  - **Incremental/compression approaches help in the short term**
  - **Preemptive migration helps further in the long term**
- **Preemptive migration has also limits**
  - **Error rate increases with lower prediction accuracy**
  - **Errors without precursor or pattern can't be predicted**
    - **Can anyone predict a non-recoverable ECC memory error?**

➲ ***Future HPC systems won't be as resilient as today's***

➲ ***Resiliency strategy for high soft error rates is missing***

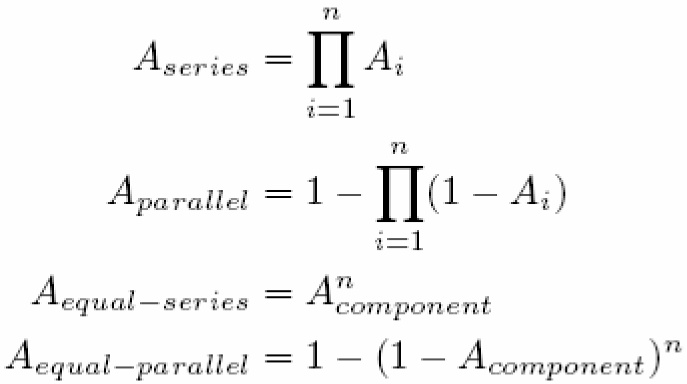
System Availability Basics
(Terms, Concepts, Models and Metrics)

- **A system's availability can be between 0 and 1, or 0% and 100%**
- **A system's availability in the long-run is based on its**
  - **Mean-time to failure (MTTF)**
  - **Mean-time to recover (MTTR)**
- **A system is rated by the number of nines in its availability metric**
- **Dependend system components are coupled serial**
- **Redundant system components are coupled parallel**
- **System components may have equal MTTF and MTTR**



$$A = \frac{MTTF}{MTTF + MTTR} = \frac{1}{1 + \frac{MTTR}{MTTF}}$$

9s	Availability	Annual Downtime
1	90%	36 days, 12 hours
2	99%	87 hours, 36 minutes
3	99.9%	8 hours, 45.6 minutes
4	99.99%	52 minutes, 33.6 seconds
5	99.999%	5 minutes, 15.4 seconds
6	99.9999%	31.5 seconds



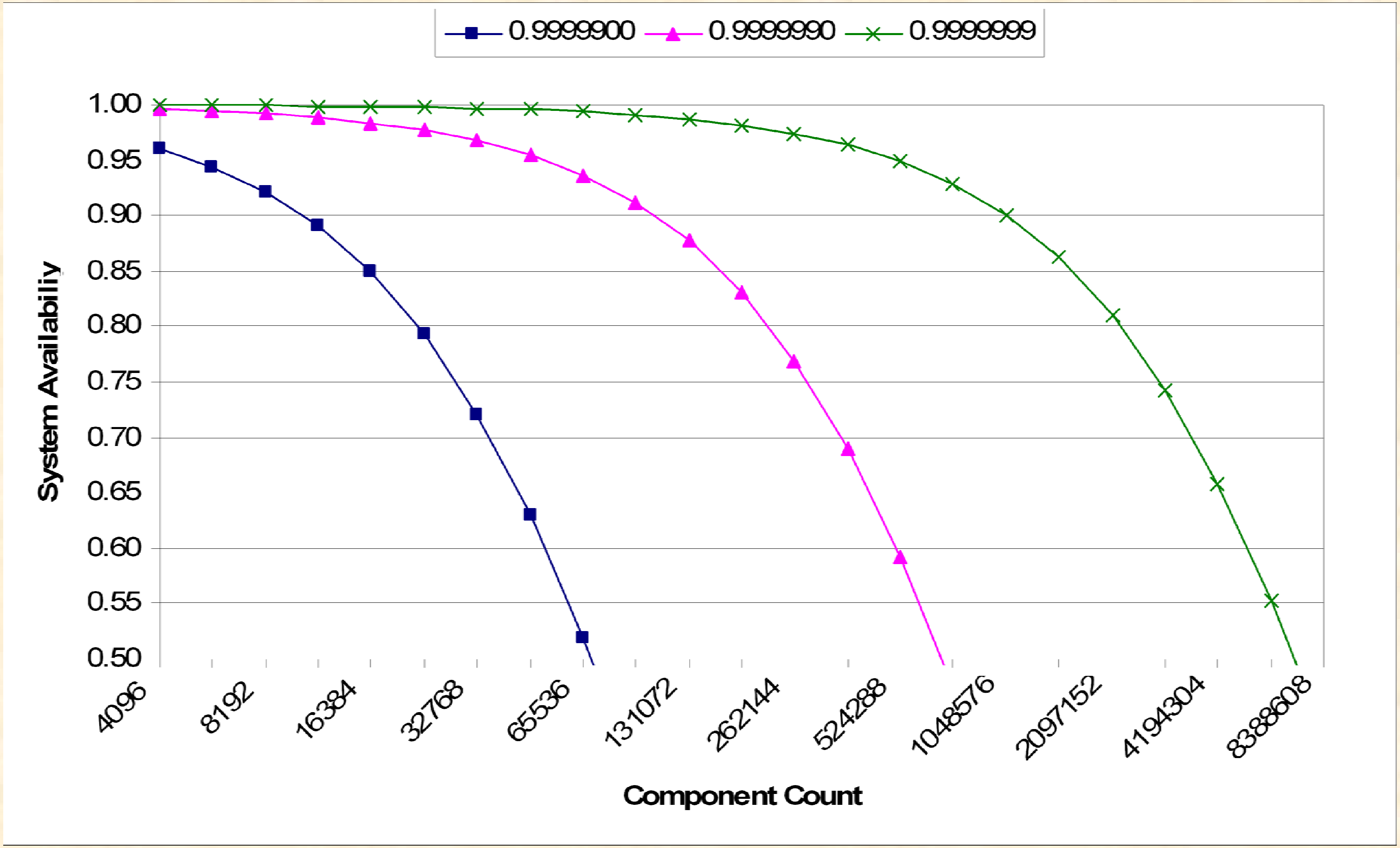
$$A_{series} = \prod_{i=1}^{n} A_i$$

$$A_{parallel} = 1 - \prod_{i=1}^{n} (1 - A_i)$$

$$A_{equal-series} = A^n_{component}$$

$$A_{equal-parallel} = 1 - (1 - A_{component})^n$$

HPC System Availability at Scale
(5, 6 and 7 Nines Compute Node Availability)



Improving System Availability with Modular Redundancy

- **Modular redundancy concepts have been around for a while**
  - **E.g. aerospace and command & control systems**

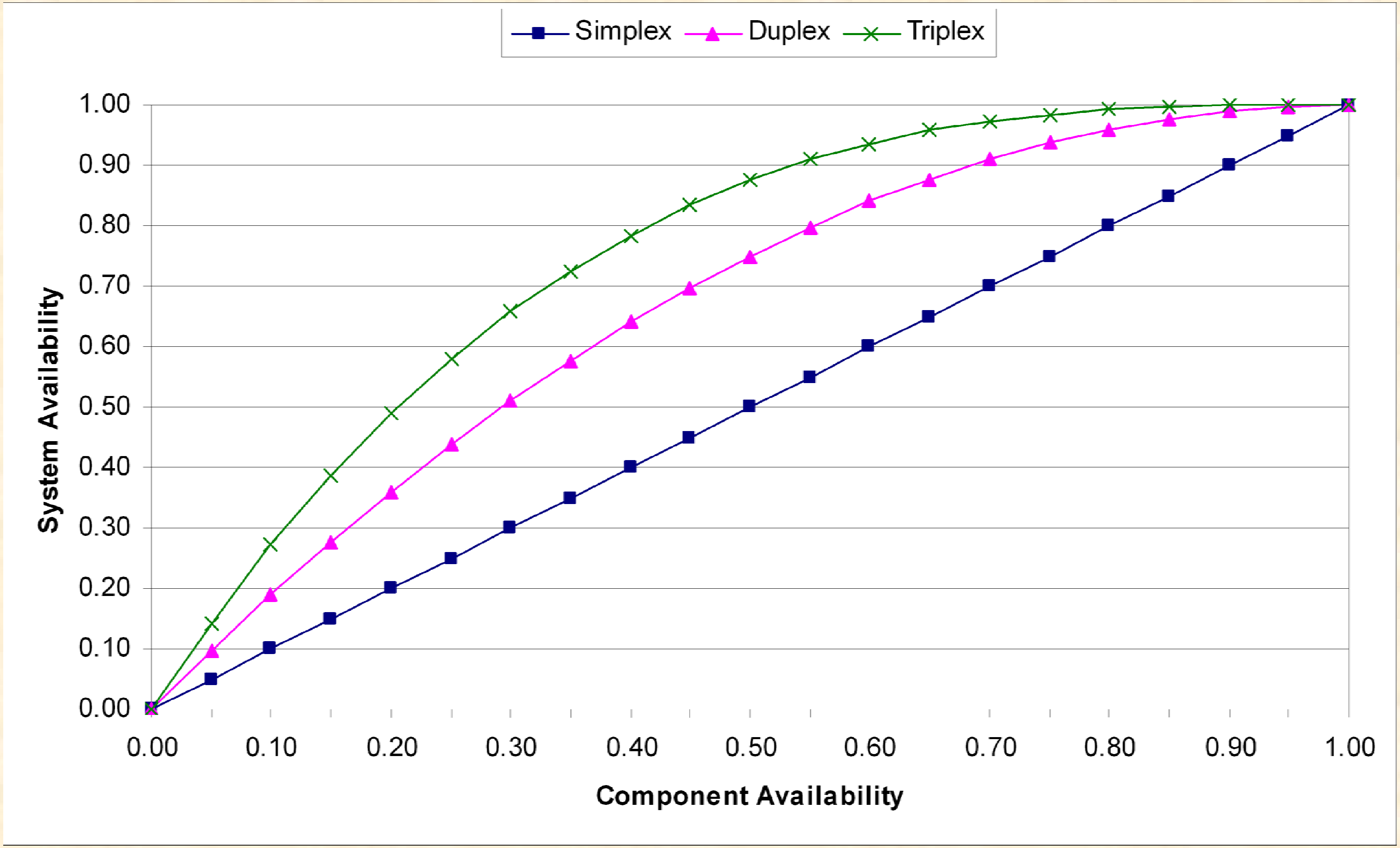
$$A_{DMR} = 1 - (1 - A)^2$$
$$A_{TMR} = 1 - (1 - A)^3$$

- **System availability is improved using redundant components**
- **Dual-modular redundancy (DMR) offers protection against hard errors and some soft errors**
- **Triple-modular redundancy (TMR) offers protection against hard and soft errors**

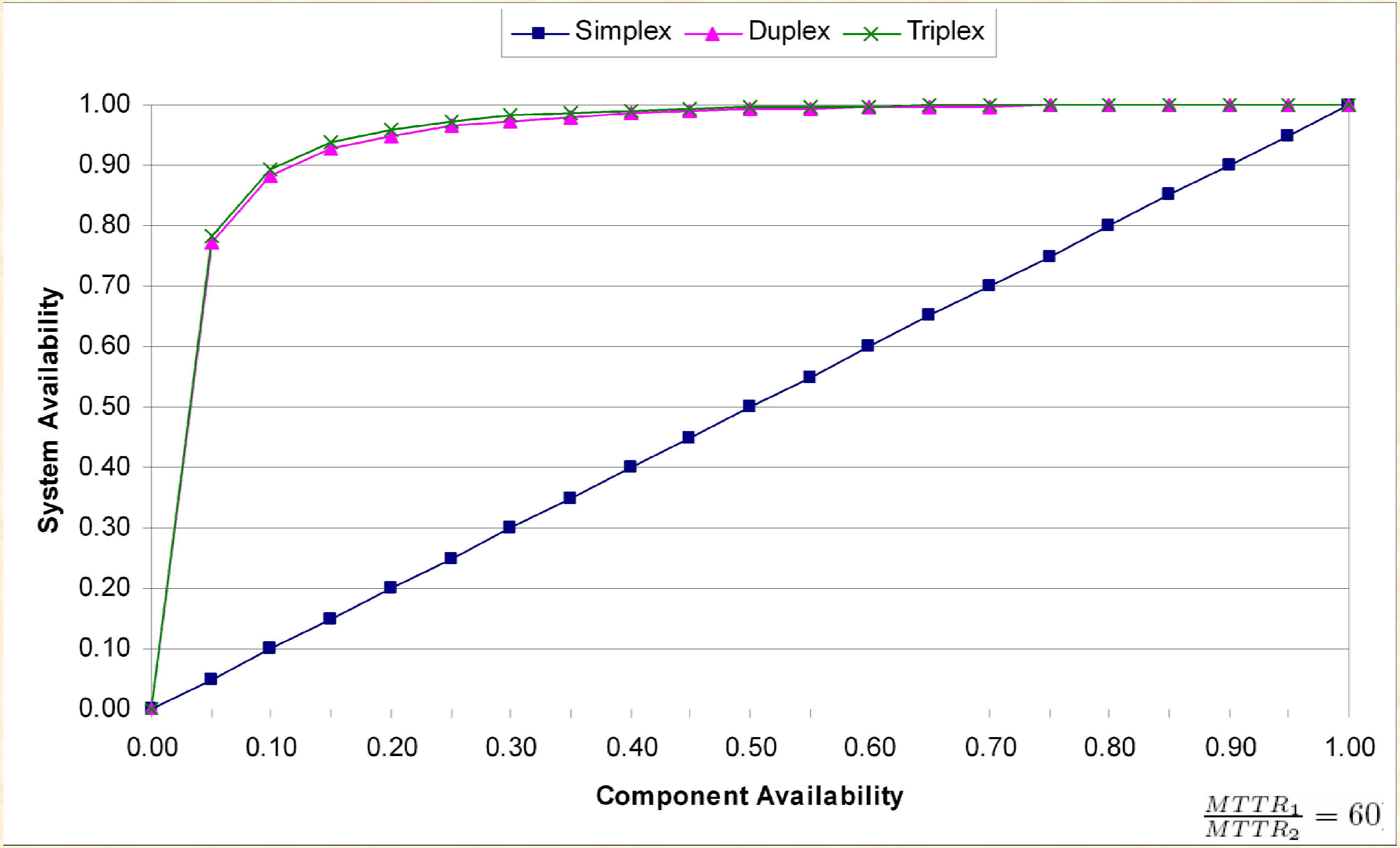
$$A_{DDMR} = 1 - (1 - A_1)(1 - A_2)$$
$$A_{DTMR} = 1 - (1 - A_1)(1 - A_2)^2$$

- **Dynamic dual- or triple-modular redundancy uses reboot or spare to reduce component MTTR**

System Availability Improvement with Modular Redundancy



System Availability Improvement with Dynamic Modular Redundancy

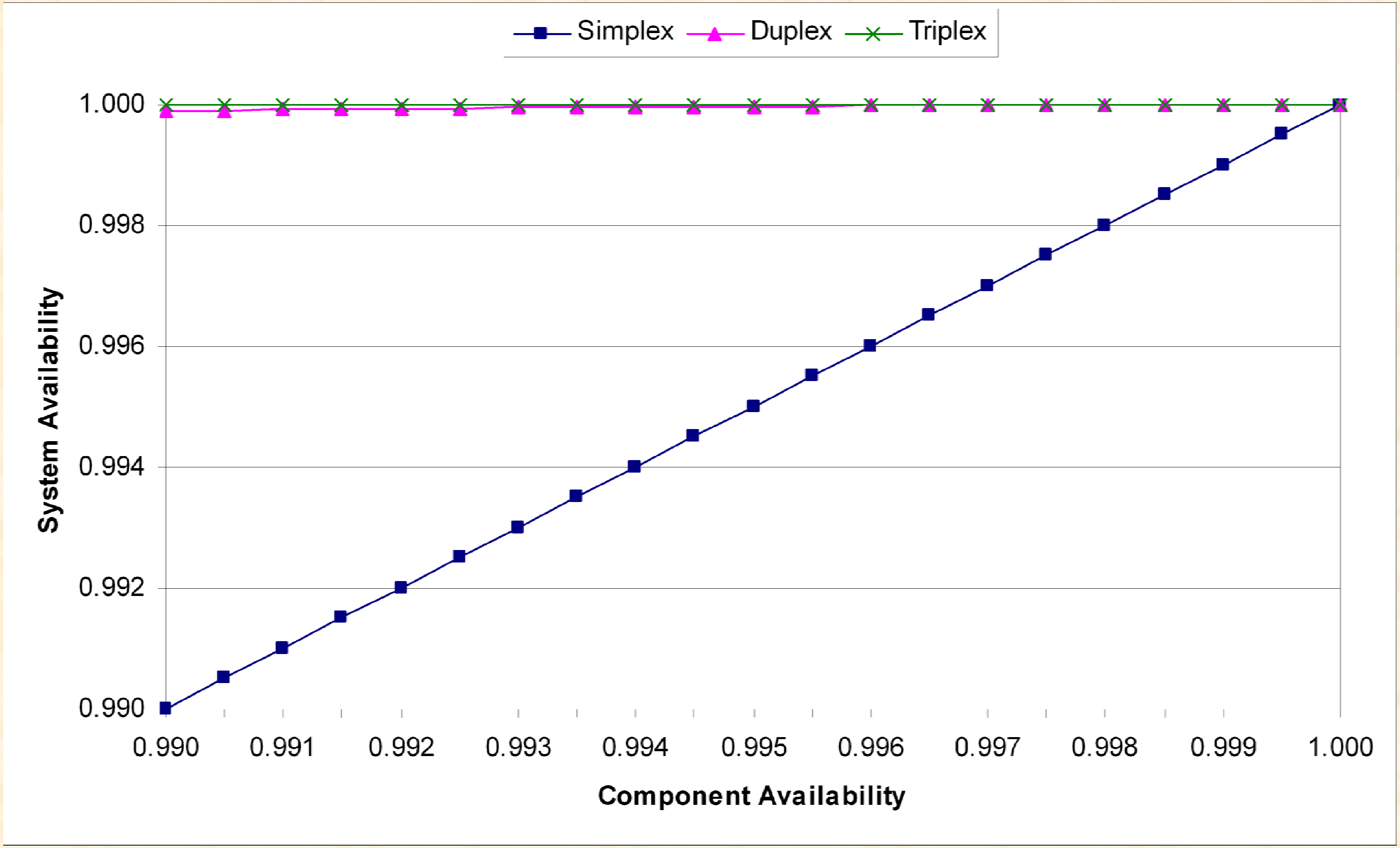


$$\frac{MTTR_1}{MTTR_2} = 60$$

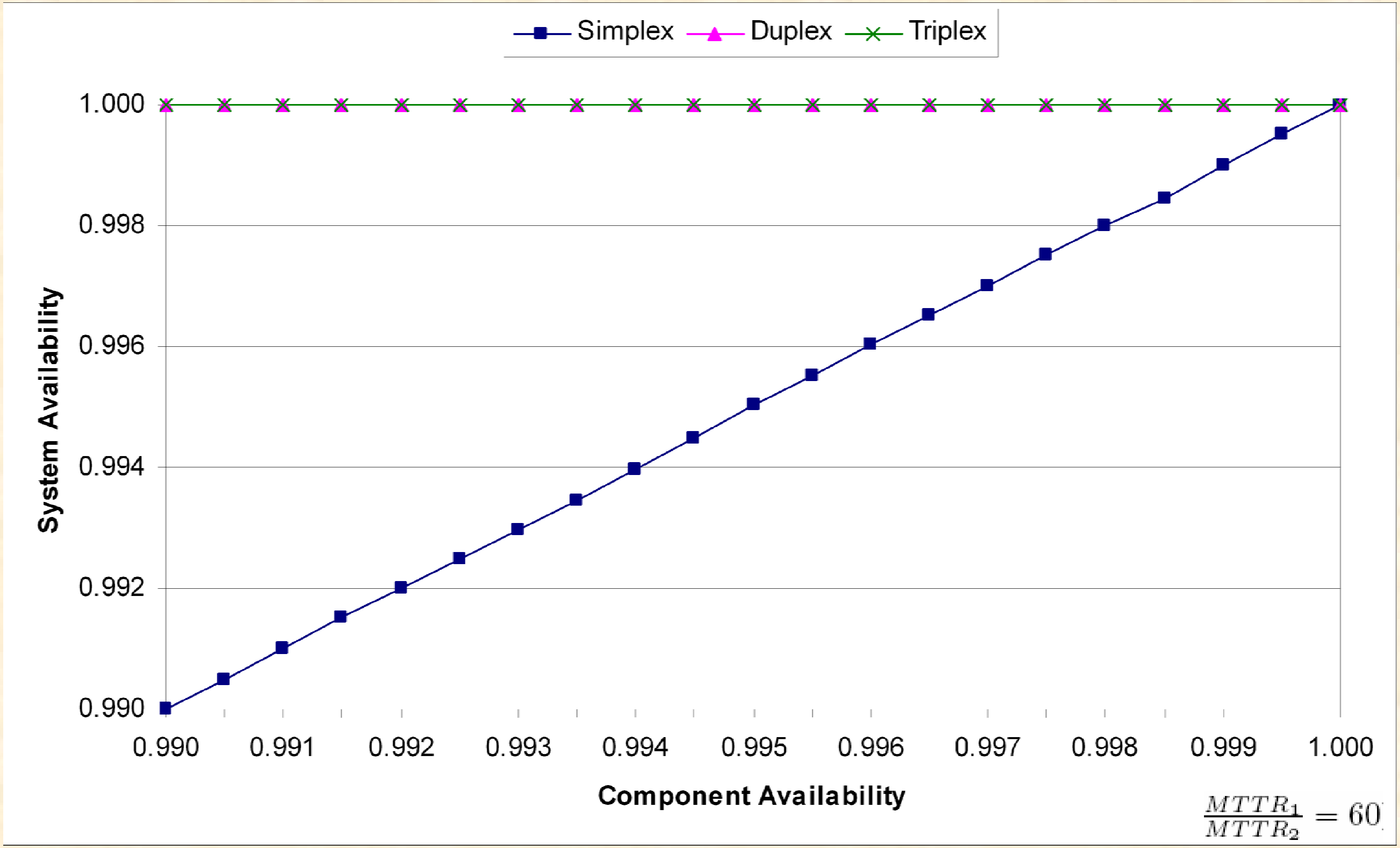
Improving Compute Node Availability with Modular Redundancy

- Today’s large-scale HPC systems have tens-to-hundreds of thousands of diskless compute nodes consisting of
  - processor(s), memory module(s) and a network interface
- Deploying modular redundancy for these systems would require to double or triple the number of compute nodes
- However, the network infrastructure is able to recover soft errors by retransmitting messages
- We only need to double or triple the number of processors and memory modules within each compute node
- A modular redundancy mechanism is needed for replication, error detection and error recovery in a massively parallel HPC system

Compute Node Availability Improvement with Modular Redundancy



Compute Node Availability Improvement with Dynamic Modular Redundancy



$$\frac{MTTR_1}{MTTR_2} = 60$$

Improving HPC System Availability with Compute-Node Modular Redundancy

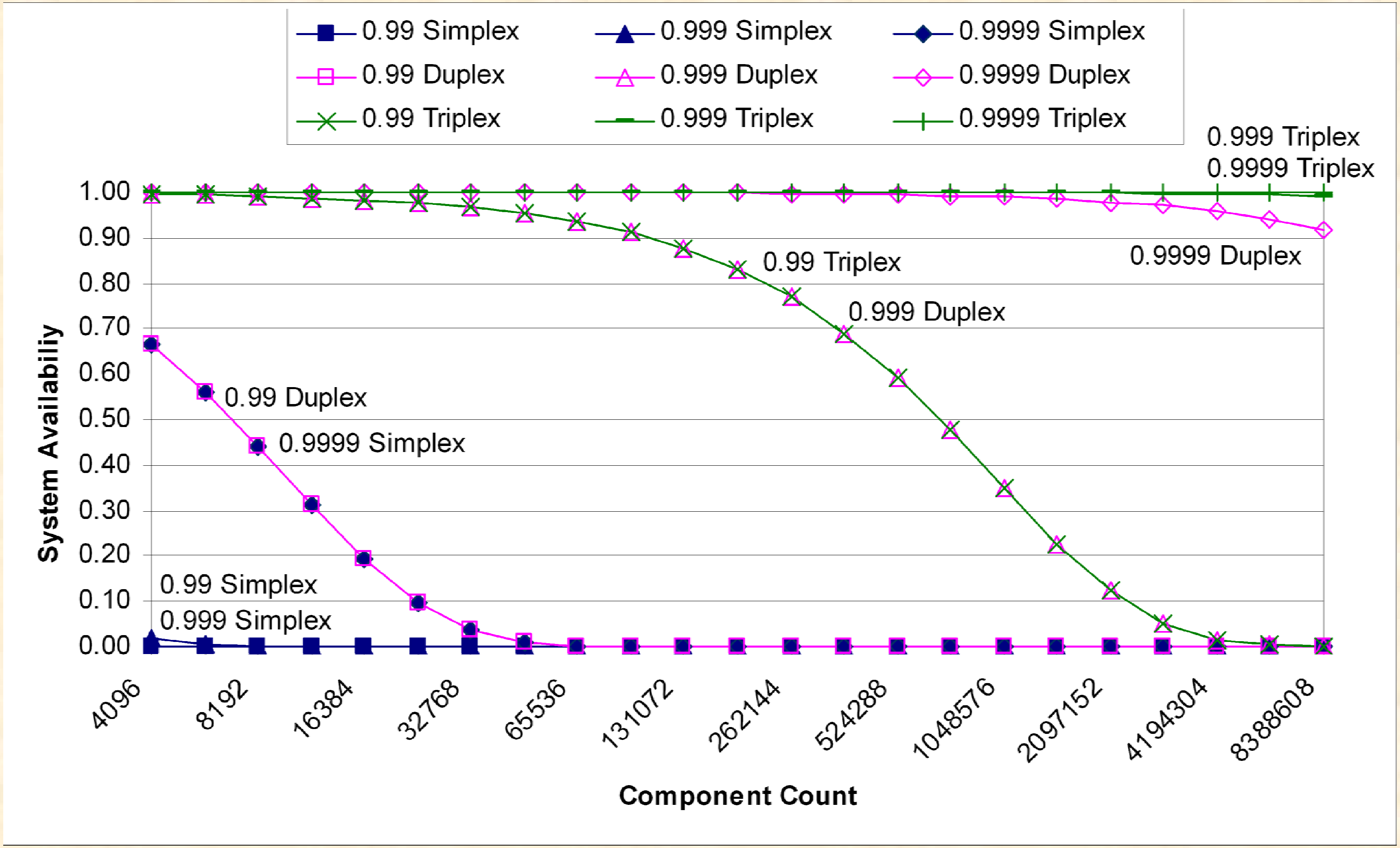
- **The availability of a modular redundant compute node is based on 2×/3× parallel coupling**
- **The availability of a HPC system is based on n× serial coupling**

$$A_{DMR} = [1 - (1 - A)^2]^n$$
$$A_{TMR} = [1 - (1 - A)^3]^n$$

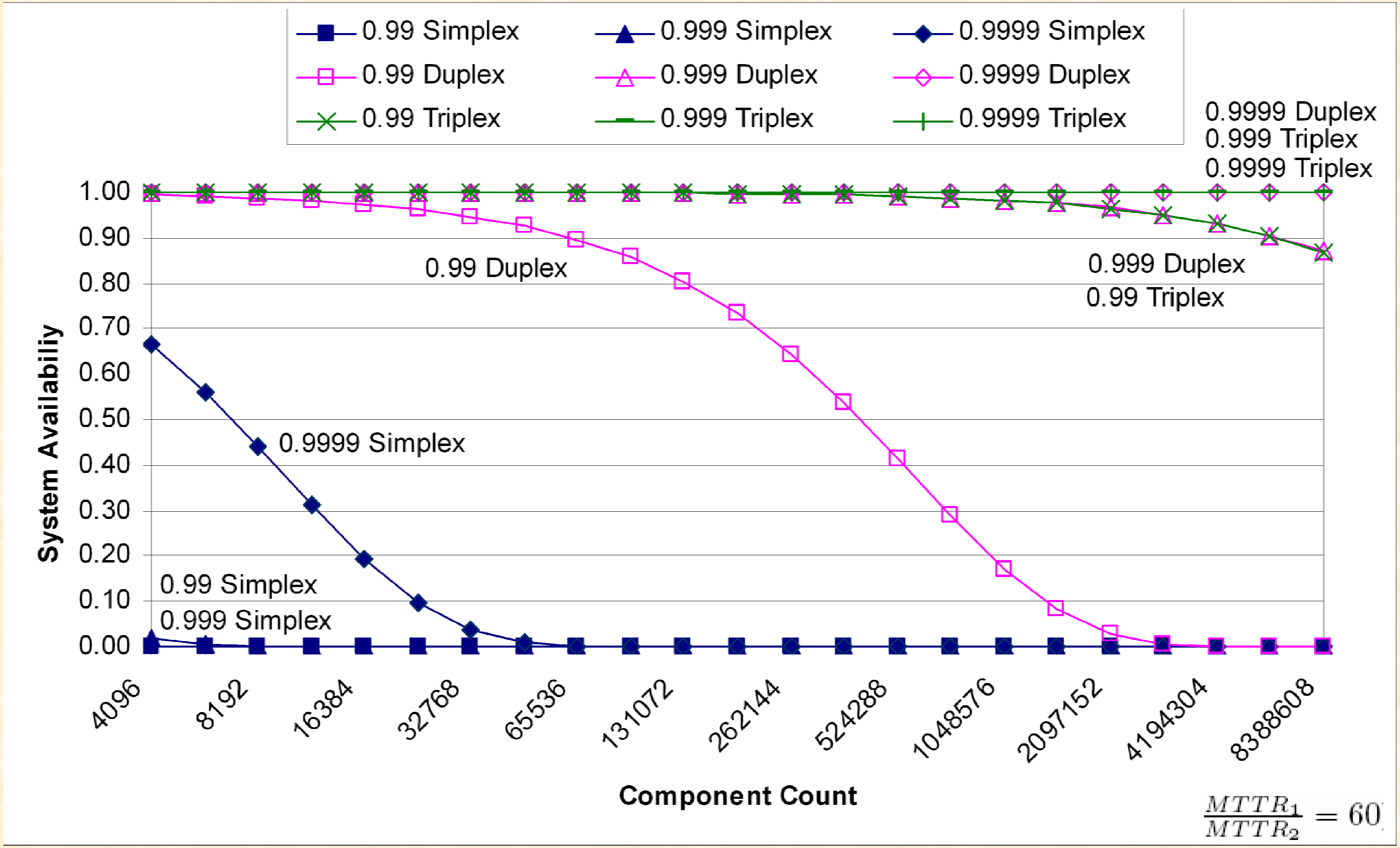
- **The availability of a compute-node modular redundant HPC system is based on n× serial of 2×/3× parallel components**
- **Dynamic modular redundancy additionally reduces the MTTR of 1 (DMR) or 2 (TMR) components**

$$A_{DDMR} = [1 - (1 - A_1)(1 - A_2)]^n$$
$$A_{DTMR} = [1 - (1 - A_1)(1 - A_2)^2]^n$$

HPC System Availability Improvement with Modular Redundancy (2, 3 and 4 Nines Compute Node Availability)



HPC System Availability Improvement with Dynamic Modular Redundancy (2, 3 and 4 Nines Compute Node Availability)



$$\frac{MTTR_1}{MTTR_2} = 60$$

Observations

- DMR and TMR for compute nodes significantly increases compute node availability, which in turn dramatically increases HPC system availability
- DMR: Compute node MTTF can be 100-1,000× less
- TMR: Compute node MTTF can be 1,000-10,000× less
- DDMR and DTMR for compute nodes improve compute node availability even further, which in turn increases HPC system availability even more
- DDMR: Compute node MTTF can be 1,000-10,000× less
- DTMR: Compute node MTTF can be 10,000-100,000× less

Conclusions

- **DMR with 4-nine or TMR with 3-nine compute node rating provides enough system availability for HPC systems planned for the next 10 years with 1,000,000 compute nodes and beyond**
- **DDMR with 3-nine or DTMR with 2-nine single component rating provides enough overall system availability for future HPC systems**
- **The reduction of individual component reliability within a modular redundant system permits recovering the costs for using 2× or 3× the number of components**
- **This tunable cost vs. reliability/availability trade-off is the counter argument to the traditional view that modular redundancy comes at 2× or 3× costs**

Conclusion and Future Work

- We have made the case for modular redundancy in large-scale HPC systems by
  - Explaining the limits for the current state of practice
  - Describing the significant increase in system availability modular redundancy offers
  - Demonstrating that modular redundancy in HPC systems allows for lowering compute node reliability and recovering the costs of using 2× or 3× the number of components
- Future work needs to focus on
  - Concepts and implementation-specific details for modular redundancy in massively parallel HPC systems
  - Mitigating the issue of increased power consumption

Questions?