Analyzing Checkpointing Trends on Petascale Systems

Harish Naik

Rinku Gupta

Pete Beckman

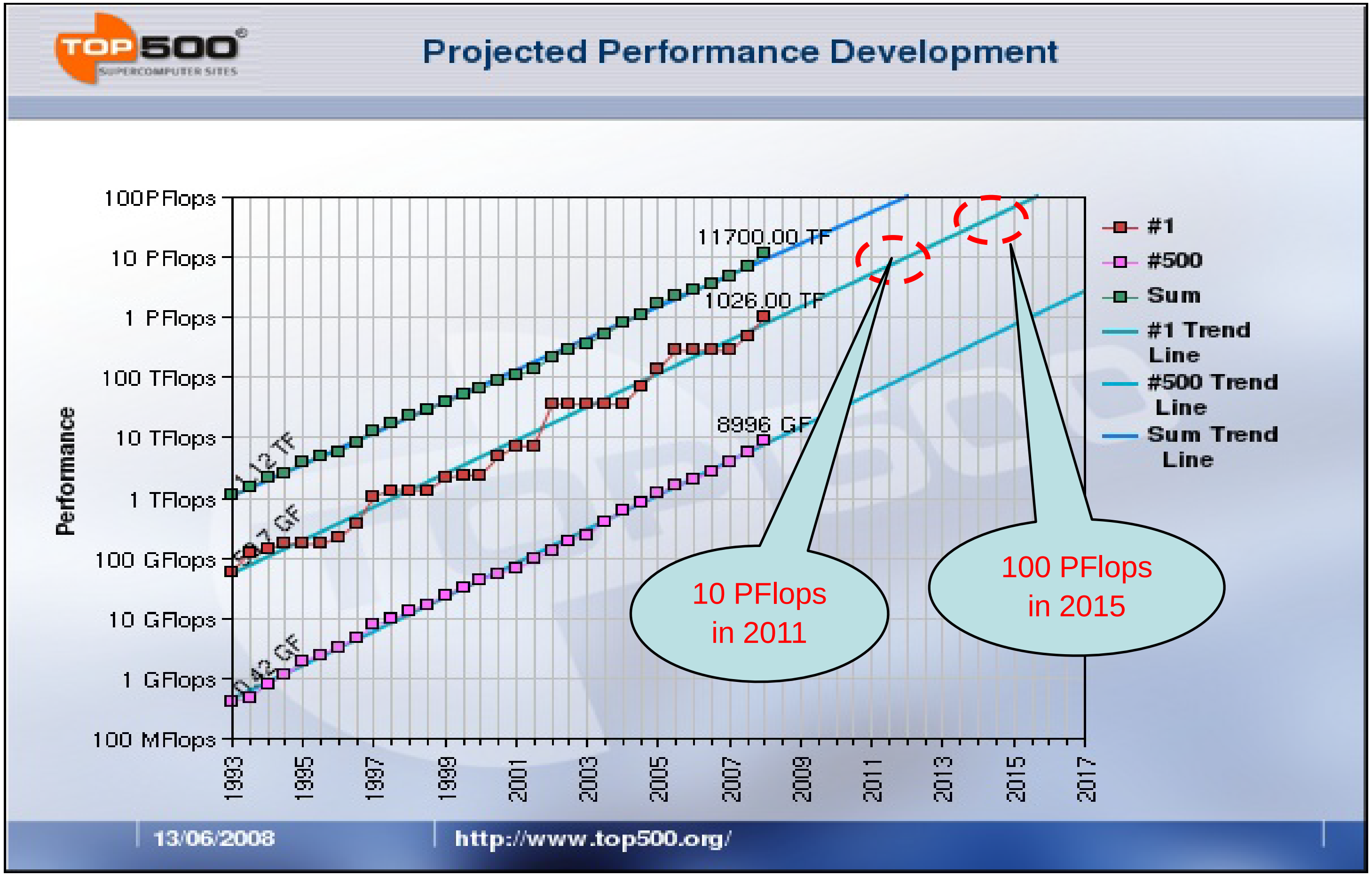
Mathematics & Computer Science Division

Argonne National Laboratory

Agenda

- Introduction
- The BG/P System
- Study and Experiments
  - Application Memory Trends
  - Checkpoint Model
- Conclusion

Introduction



- As systems increase in size, large-scale faults become unavoidable

Fault Tolerance and Checkpointing

- Wide variety of research has focused on Fault Tolerance
  - Focus on hardware as well as software
  - Focus on different levels of high-end computing software stack
- Checkpointing and Recovery
  - Popular and widely accepted method
    - Checkpointing: Involves periodically saving state to storage
    - Recovery: Involves rolling back to a previously saved state
- Emerging machines pose new challenges for this popular method
  - Limited network resources, limited I/O bandwidth

Checkpointing on Petascale supercomputers

- Important Questions
  - How feasible is it to checkpoint applications on modern machines?
    - Challenges exposed for checkpointing by large leadership machines are *very* different and on different scale
  - How much time should user devote to checkpointing?
    - What % of cost should be devoted for fault tolerance?
  - Can the user intelligently decide when and where to checkpoint?

Research Focus

- Focus of this research
  - Understands memory trends of popular supercomputing applications
    - With Focus on the IBM BG/P supercomputer at Argonne National Laboratory
  - Presents an analytical model for computing checkpoint frequencies and limitations to assist end-users

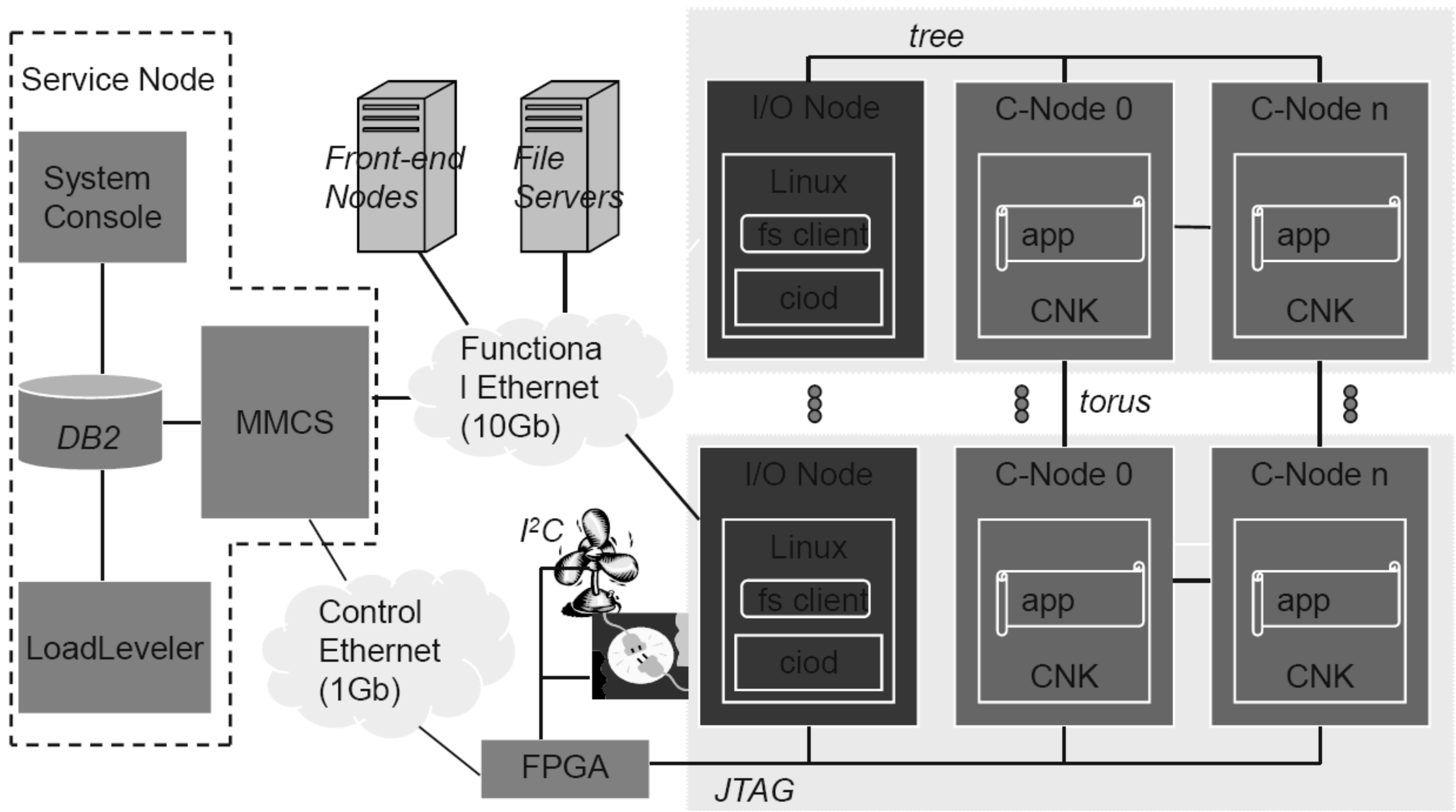
Agenda

- Introduction
- **The BG/P System**
- Study and Experiments
  - Application Memory Trends
  - Checkpoint Model
- Conclusion

The Blue Gene/P 'Intrepid' System at Argonne National Laboratory

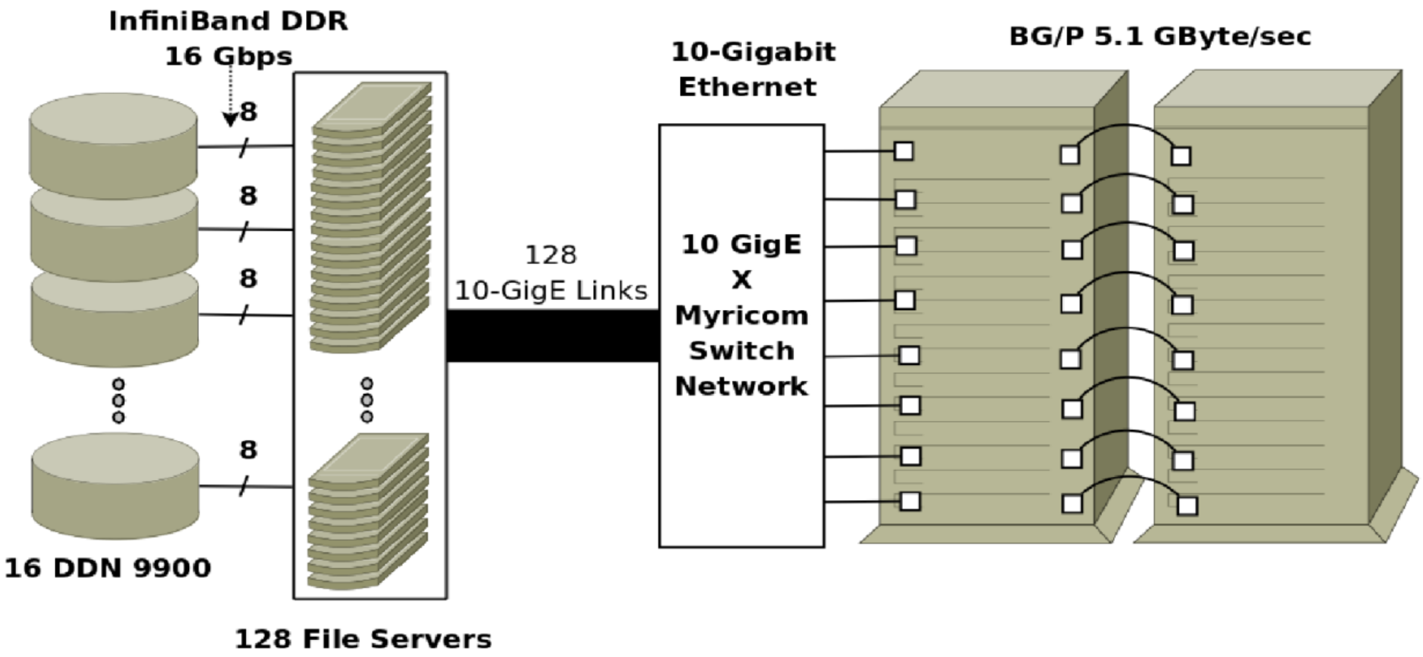
- Brief Description
  - Peak performance: 556 TF
  - 40 rack machine
  - Each rack has 1024 nodes (40,960 nodes)
    - *Each node has 4 cores (163,840 cores)*
  - 80 TB of Memory
  - Compute Nodes run a light weight OS called Compute Node Kernel (CNK)
  - 640 I/O nodes to communicate with the file system
  - I/O nodes and Compute Nodes are in the ratio 1:64
  - Login nodes for front end tasks like compiling etc.

'Intrepid' Architecture



- Supports 5 different networks
- Torus network available for application communication
- 10-Gigabit Ethernet network connects I/O nodes, file servers and storage devices

BGP at Argonne - File system



- Backend Storage
  - 16 DataDirect 9900 SAN storage arrays (8 PetaBytes raw storage)
    - Each DDN connects to 8 file servers through 8 DDR InfiniBand links
    - ‘Intrepid’ system consists of 128 such file servers
  - Each file server connects to Myricom 10GbE switching network through a 10GbE link
- Each I/O node also connects to Myricom 10GbE network through a 10GbE link
  - Peak bandwidth is only 6.8Gbps from each I/O port

Checkpointing Techniques

- Application level (or user-defined) checkpointing
  - Checkpoints are intelligently placed
  - More programmer effort
  - Portable since the checkpoints defined in machine independent format inside the application
  - *Library provided by IBM for BG/P; library exposes a small API that can be used by end-users*

- Operating-System level checkpointing
  - User transparent way
  - Entire application state is saved
  - OS has no idea about the structure and data inside the application. Hence total size of saved data is huge.
  - On petascale systems, this can lead to tremendous I/O overhead with increasing system size
  - Not supported on IBM BG/P

Checkpointing Optimizations

- Full Memory checkpointing
  - Entire memory context for the process is saved during each checkpoint
  - *IBM BG/P Checkpoint library supports full checkpointing*
  - *Focus of our current study*

- Incremental checkpointing
  - Saves only modified pages since last checkpoint
  - Can reduce memory context; can be useful for large systems
  - Not supported on IBM BG/P currently
  - Focus of our future work

Agenda

- Introduction
- The BG/P System
- **Study and Experiments**
  - **Application Memory Trends**
  - **Checkpoint Model**
- Conclusion

Applications on the BG/P

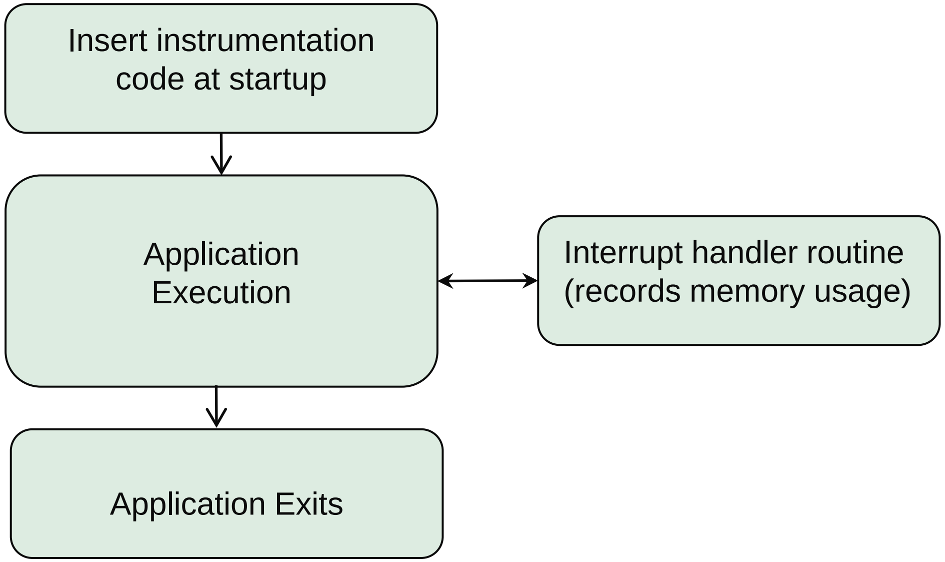
- Computational Fluid Dynamics Application
  - NEK5000
    - *Developed at Argonne National Laboratory; Gordon Bell Prize winner*
    - *Spectral element multigrid solver coupled to a highly scalable, parallel coarse grid solver*
    - *Highly scalable for over 100K cores; used by many research organizations worldwide*

- Molecular Dynamics Simulations
  - MD Density Functional theory (DFT) applications chosen due to their stringent computational demands and memory requirements
    - ***Grid-based Projector-Augmented Wave (GPAW)***
    - ***Carr-Parrinello Molecular Dynamics (CPMD)***

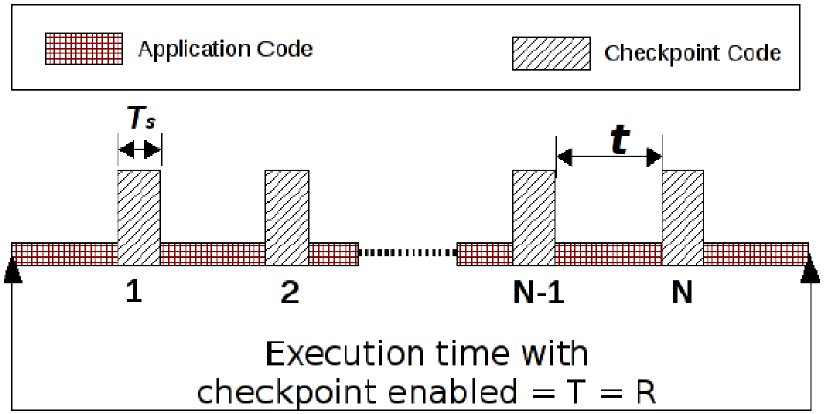
Methodology

- Step 1:
  - Understand memory usage and trends
    - *change over application execution time*
    - *change with system size*
  - Memory usage measured using timers at regular intervals
    - *Timer uses the getrusage function for memory usage measurement*

- Step 2:
  - Plugging memory usage trends in optimal checkpoint model
    - *Helps end-user estimate checkpoint frequency*
    - *And checkpointing interval*



Simple Optimum Checkpoint Model (1)



- Majority of scientific applications have constant memory pattern over majority of their execution lifetime
- **T** → Total time of application execution including checkpoints (can be approximated to reservation time R)
- $\mathbf{T_s}$ → Time required to complete one full checkpoint
- **N** → Optimum number of checkpoints to be performed
- **t** → Optimum interval between checkpoints

Simple Optimum Checkpoint Model (2)

- Checkpoint Model derives number of optimum checkpoints based on
  1. Percentage of reservation time (or runtime) dedicated for checkpointing,
  2. Bandwidth from the compute node to file servers and
  3. The total amount of data to be checkpointed

Thus,

- **B** → Unidirectional bandwidth from compute nodes to storage disks
- **X%** → Percentage of time user is willing to spend performing checkpointing
- **n** → Number of cores that the application is run on
- **M** → Mean memory usage per core

..we can deduce that

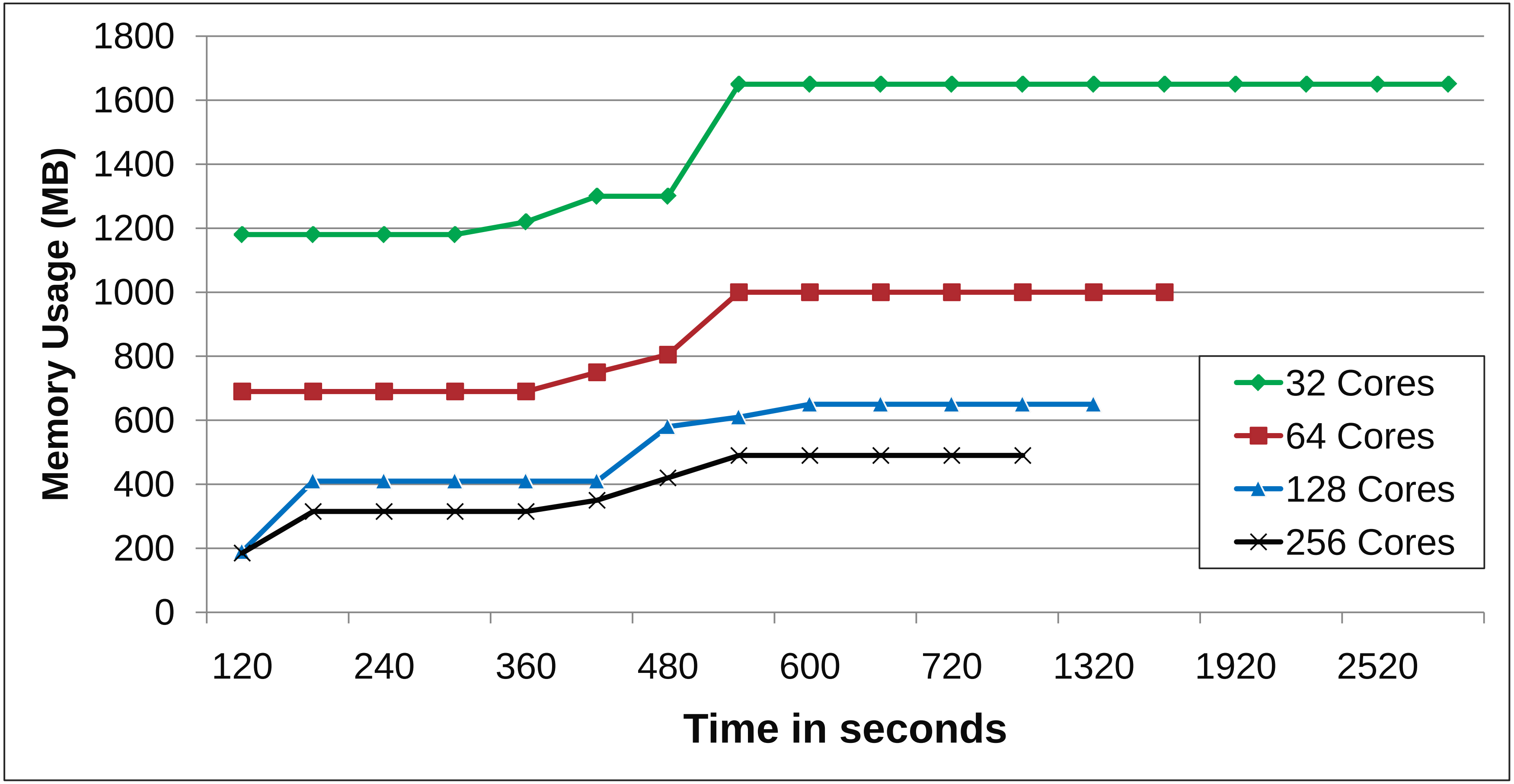
N (i.e. number of checkpoints) **= lower bound (XRB/nM)**

t (time interval between two checkpoints) **= M (n/X - 1)/B**

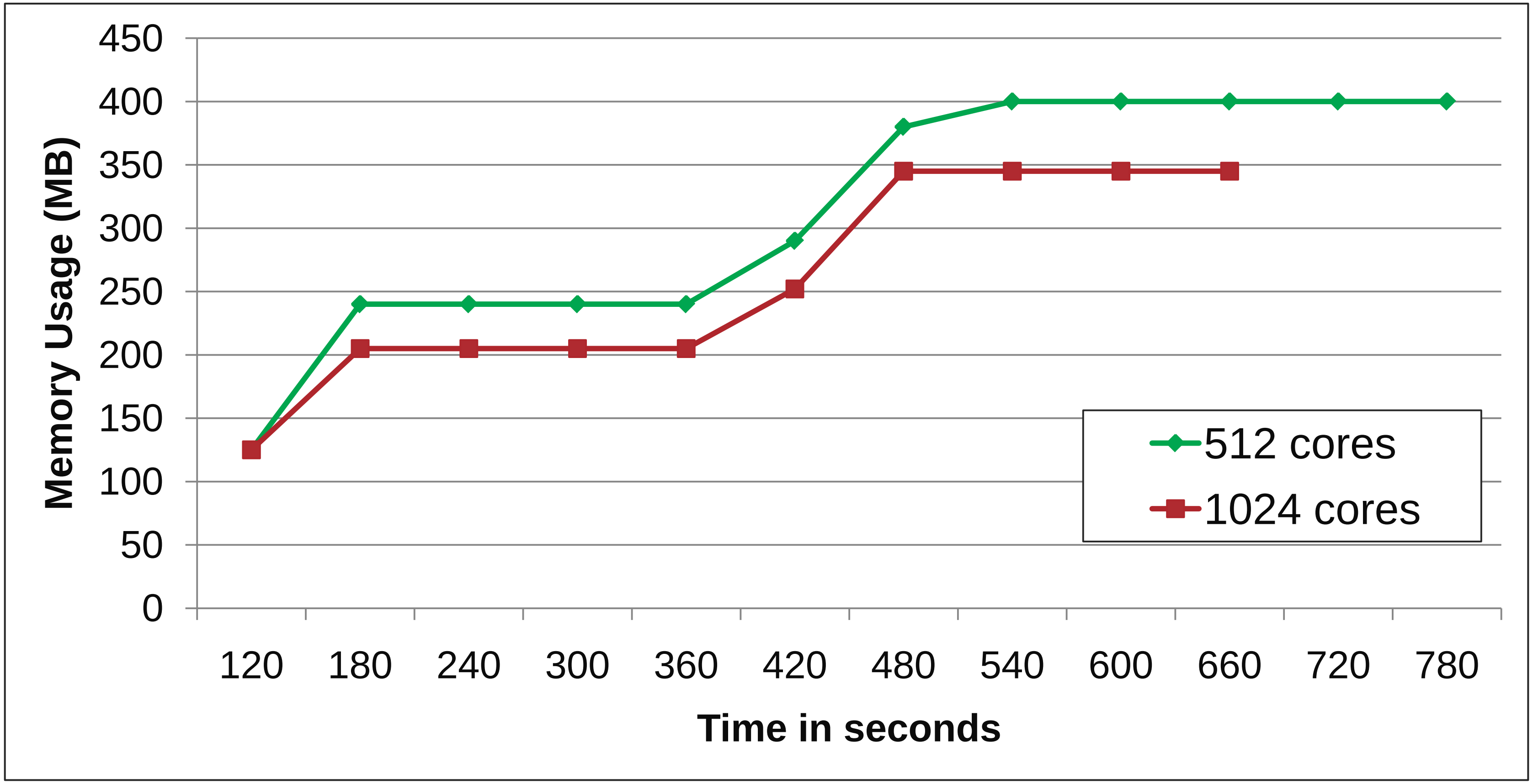
Challenges

- Accurate prediction of 'B' is difficult due to resource sharing; however users can make educated guesses
- Reservation time 'R' can be different from application run time; however assumptions can be made based on historical data and past runs

GPAW Memory Consumption (1)



GPAW Memory Consumption (2)



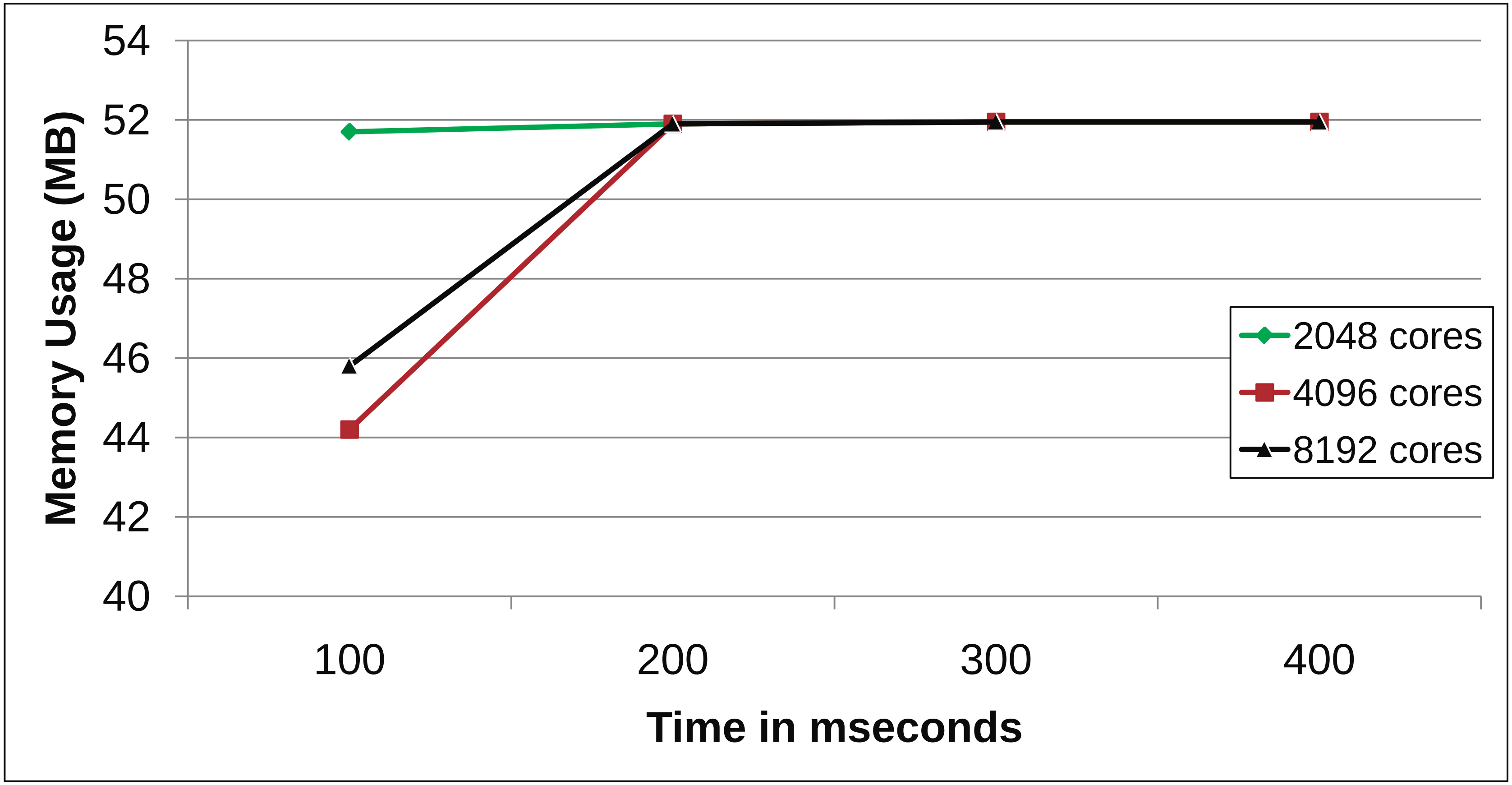
Computed Values for GPAW

n	M	T_A	X	30% Bandwidth			60% Bandwidth		
				B_{30}	N_{30}	t_{30}	B_{60}	N_{60}	t_{60}
32	1650	3168	0.3	127.5	2	1584	255	4	792
64	1000	1914	0.3	255	2	957	510	4	478.5
128	650	1386	0.3	510	2	693	1020	5	277.2
256	475	990	0.3	1020	2	495	2040	4	247.5
512	400	858	0.3	2040	2	429	4080	5	171.6
1024	350	726	0.3	4080	2	363	8160	4	181.5

- The GPAW application is executed in SMP mode with one thread, with only core being used on each compute node

- The total bandwidth available to the GPAW application can be computed by: (n/64)*Bi/o)

CPMD Memory Consumption

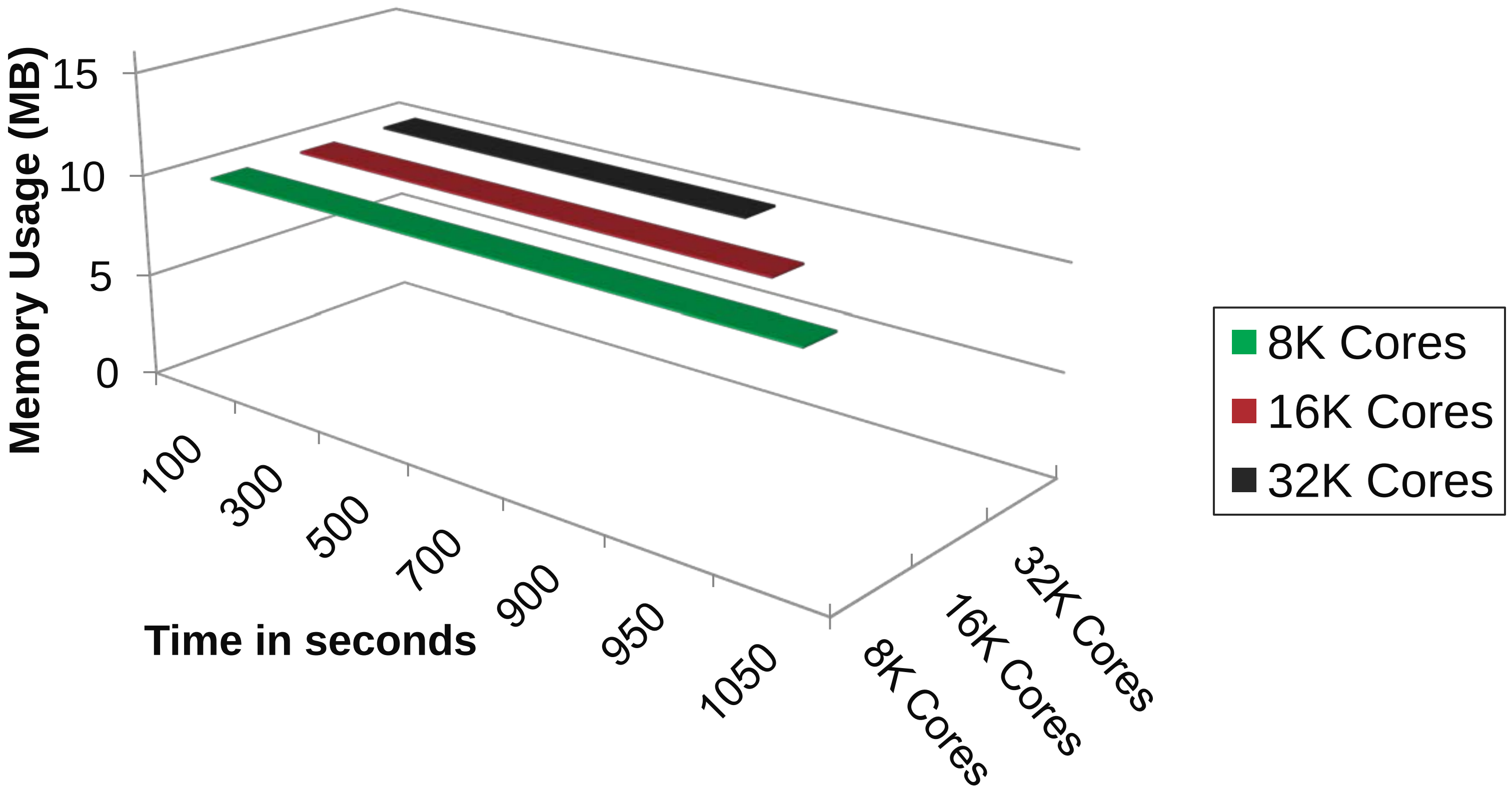


Computed Values for CPMD

n	M	T_A	X	30% Bandwidth			60% Bandwidth		
				B_{30}	N_{30}	t_{30}	B_{60}	N_{60}	t_{60}
2048	51.82	220	0.4	2040	1	220	4080	3	73.33
4096	51.85	330	0.4	4080	2	165	8160	5	66
8192	51.87	440	0.4	8160	3	146.67	16320	6	73.33

- The NEK5000 application is executed in SMP mode with 4 threads
- The total bandwidth available to this application can be computed by: (n/(64*4))*Bi/o)

NEK5000 Memory Consumption



Computed Values for NEK5000

n	M	T_A	X	30% Bandwidth			60% Bandwidth		
				B_{30}	N_{30}	t_{30}	B_{60}	N_{60}	t_{60}
4096	25.13	960	0.07	4080	2	480	8160	5	192
8192	25.13	960	0.07	8160	2	480	16320	5	192
16384	25.13	900	0.07	16320	2	450	32640	4	225
32768	23.57	900	0.07	32640	2	450	65280	5	180
65536	23.57	900	0.07	65280	2	450	130560	5	180

- The NEK5000 application is executed in virtual mode
- The total bandwidth available to this application can be computed by: (n/(64*4))*Bi/o)

Agenda

- Introduction
- The BG/P System
- Study and Experiments
  - Application Memory Trends
  - Checkpoint Model
- **Conclusion**

Conclusions

- Study to show memory trends of popular applications on Blue Gene/P supercomputer
  - Memory trends allow end-users estimate amount of time needed for checkpointing
  - Considered *full checkpointing* where entire program state is saved
    - This model was chosen since IBM checkpointing library supports only full checkpointing at this point
- Presented an analytical model for computing checkpoint frequencies and intervals
  - Studied applications and computed values based on the model
- Showed how application scaling influences checkpoint-related decisions
- Future work consists of conducting similar study for incremental checkpointing; studying larger-scale applications and measuring checkpointing time

Questions?