Towards an Exa-scale Operating System*

Ely Levy, The Hebrew University

*Work supported in part by a grant from the DFG program SPPEXA, project FFMK

The eXascale challenge

1,000,000 = 1M teraflop (double precision)

Low-power ~20MW (~10% of current systems).

- Current top CPUs/GPUs/Phi: ~O(1) teraflops, ~O(10) integer operations.
- **To obtain 1M teraflop we need:**
  - ~1M CPU for traditional applications.
  - ~ 100K processors for integer applications.

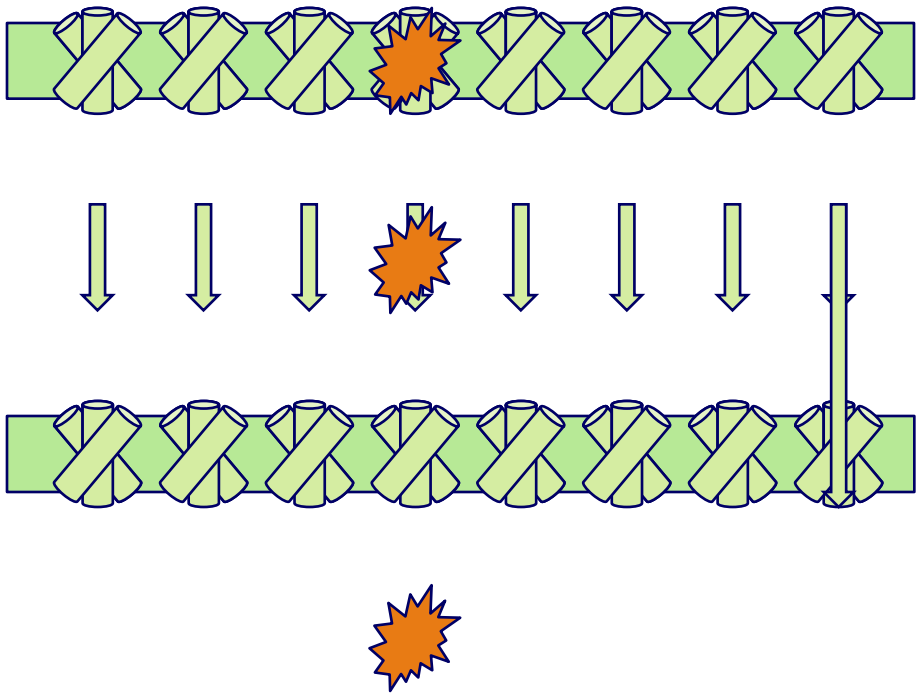
The project in a Nutshell

- **Exascale challenges include:**
  - Scaling
  - Failures
  - Load imbalances
  - Heat and power management
  - Information collection
- **Self-organizing platform and applications**
- **Adapt and combine 4 mature technologies**
  - L4(micro kernel), XtreemFS, MOSIX and MPI

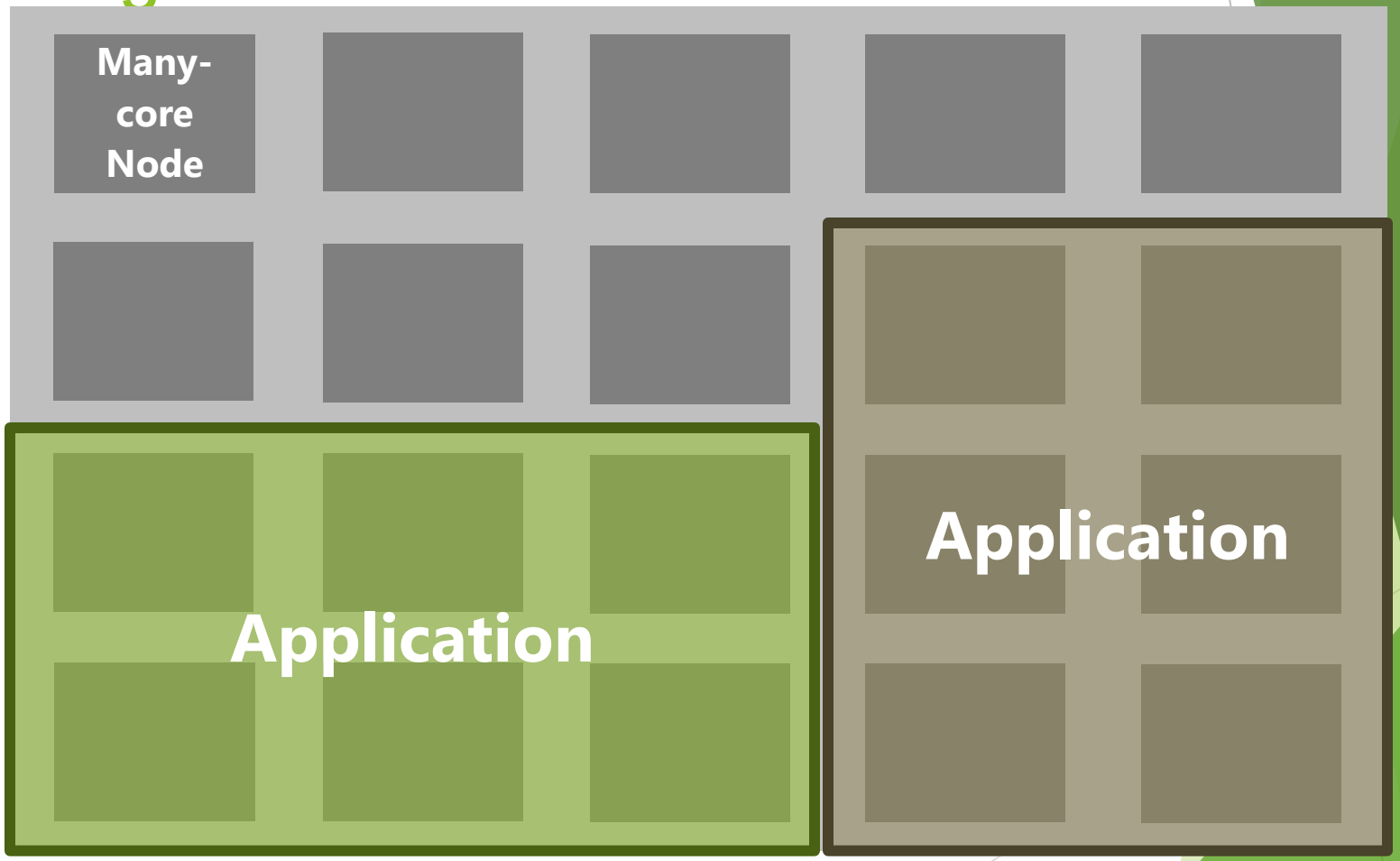
Hardware assumptions

- **Large numbers of components**
  - High failure rates
  - Not all cores may be active simultaneously due to heat/energy constraints
- **Low-power storage for checkpoint state on each node**
- **Usage:**
  Manycore Nodes with compute and service cores

Target Applications

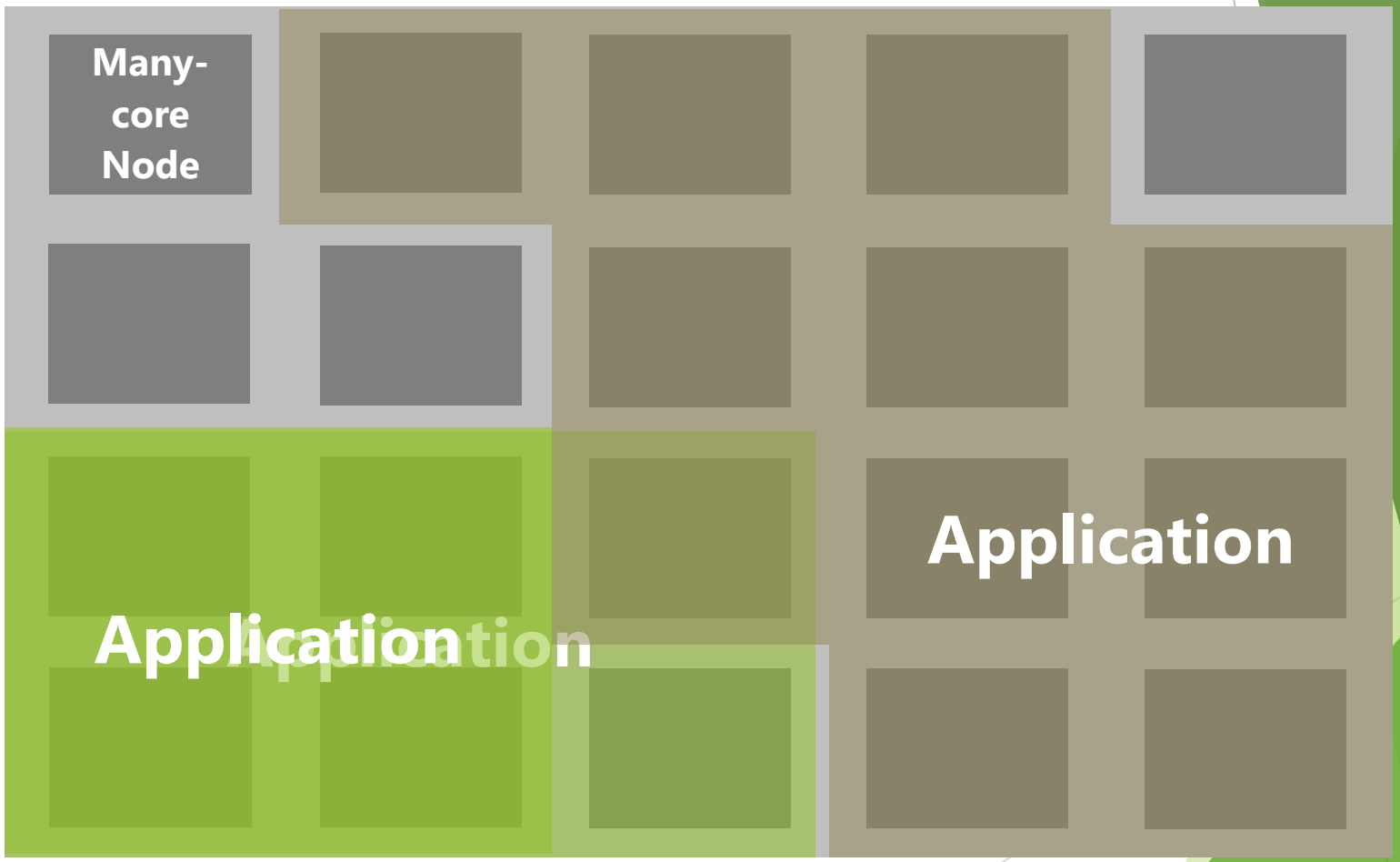


State of the Art: Static Assignment



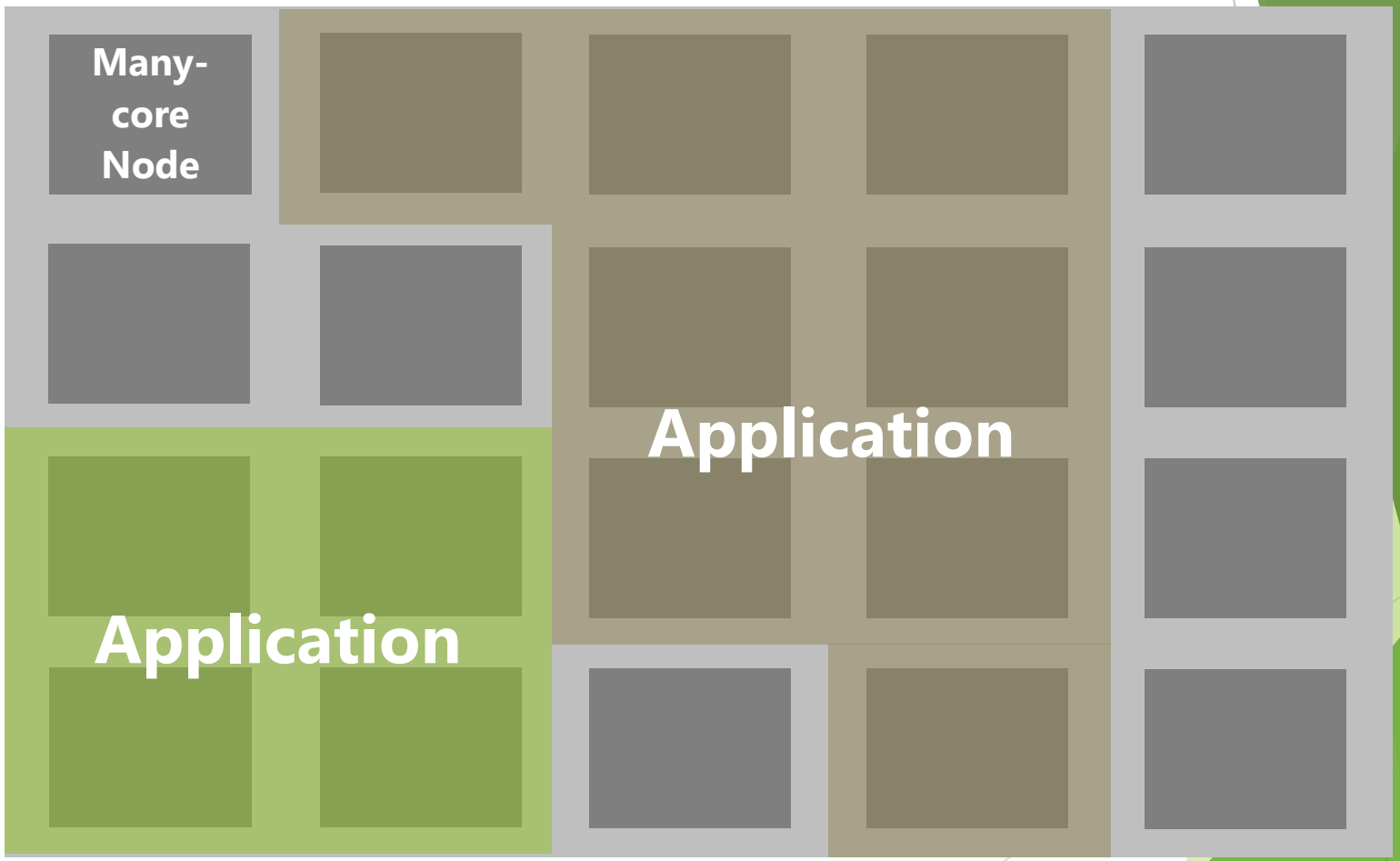
Goal - Dynamic Applications

Exascale System



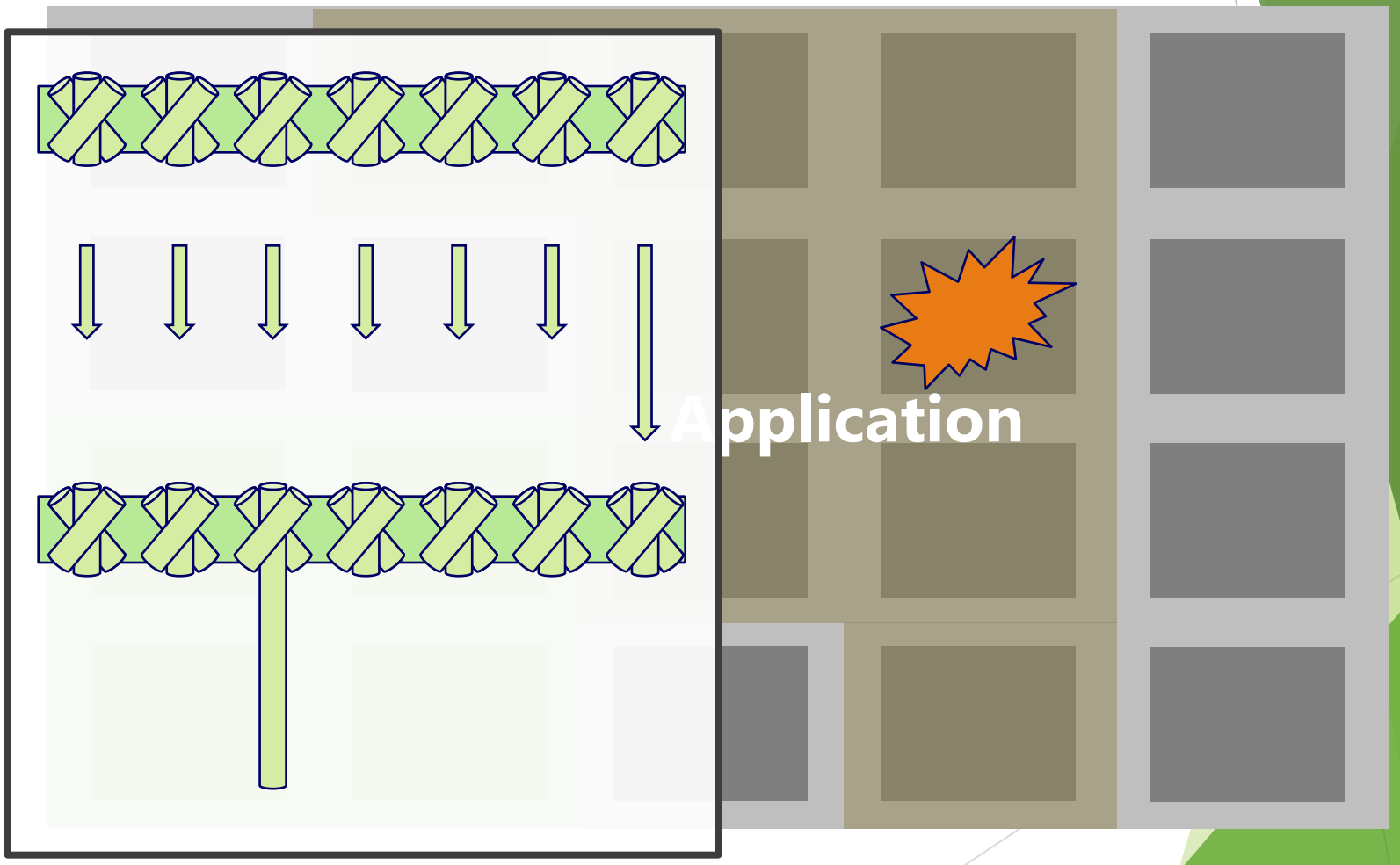
Goal - Dynamic Applications

Exascale System

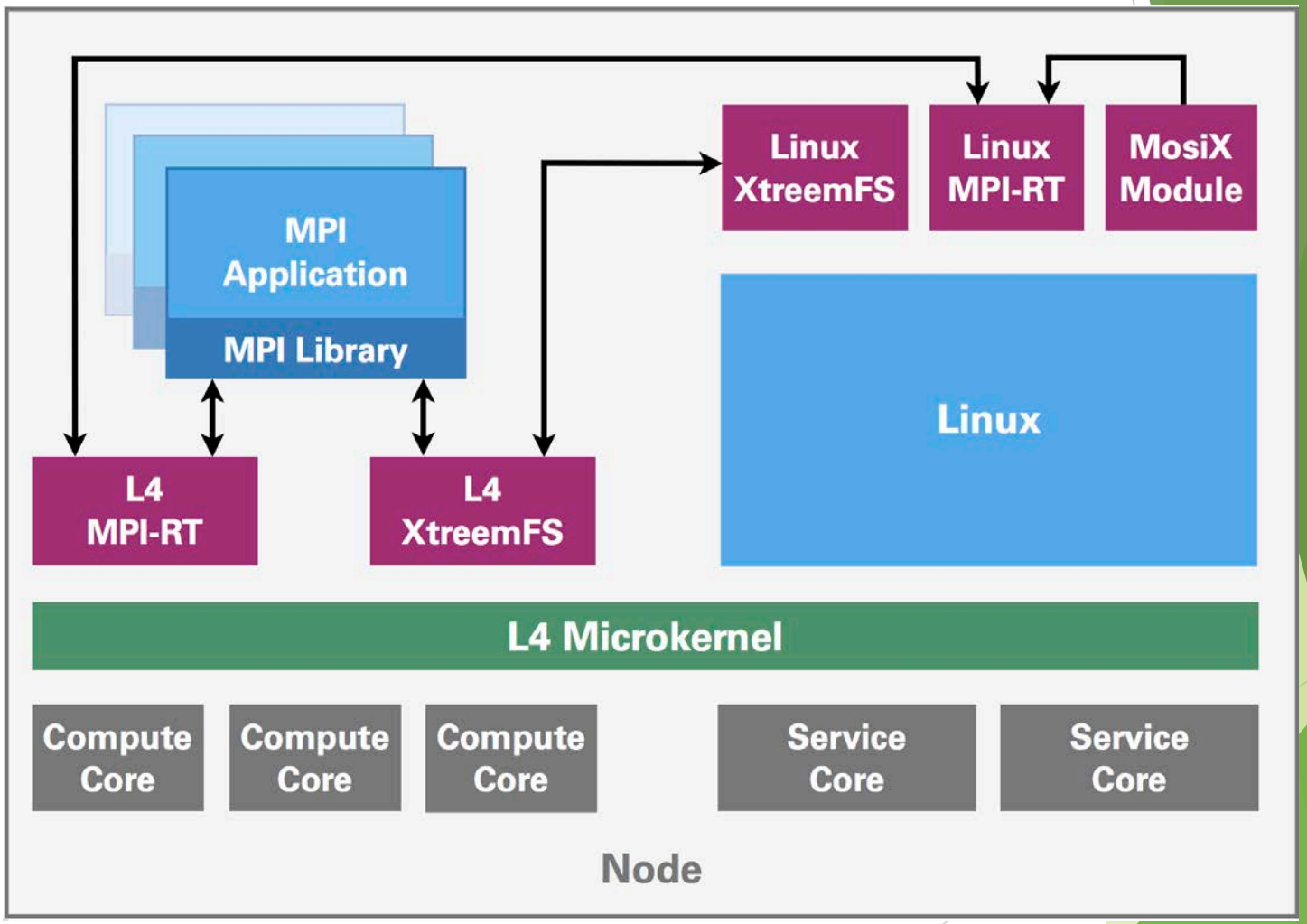


Challenges

Exascale System



System Architecture



The information collection problem

Given a cluster with O(1M) nodes:

- **Nodes: computing servers or mobile devices.**
  - Each node sends 1 message each unit of time.
  - Message contains information about the state of the node and its relevant resources, e.g., availability, load, free memory, temperature, etc.
- **One master computer regularly collects information about the state of all the nodes.**
  - Performs management decisions that require system-wide info, e.g., job allocations as in MPI or SLURM, load-balancing, IPC optimization.
  - Can be mirrored for fault tolerance.

The problem: how to collect fresh information without overloading the master computer.

Distributed bulletin board

- **Information is circulated continually.**
  - Unit of time not too small - reduces communication congestion.
  - Recall that some events, e.g. load-balancing, are triggered by cluster nodes, not by the master computer.
- **Available instantly to client processes, even with "relaxed" circulation.**
  - Example: watch – works continuously, provides the time instantly.

Possible algorithms

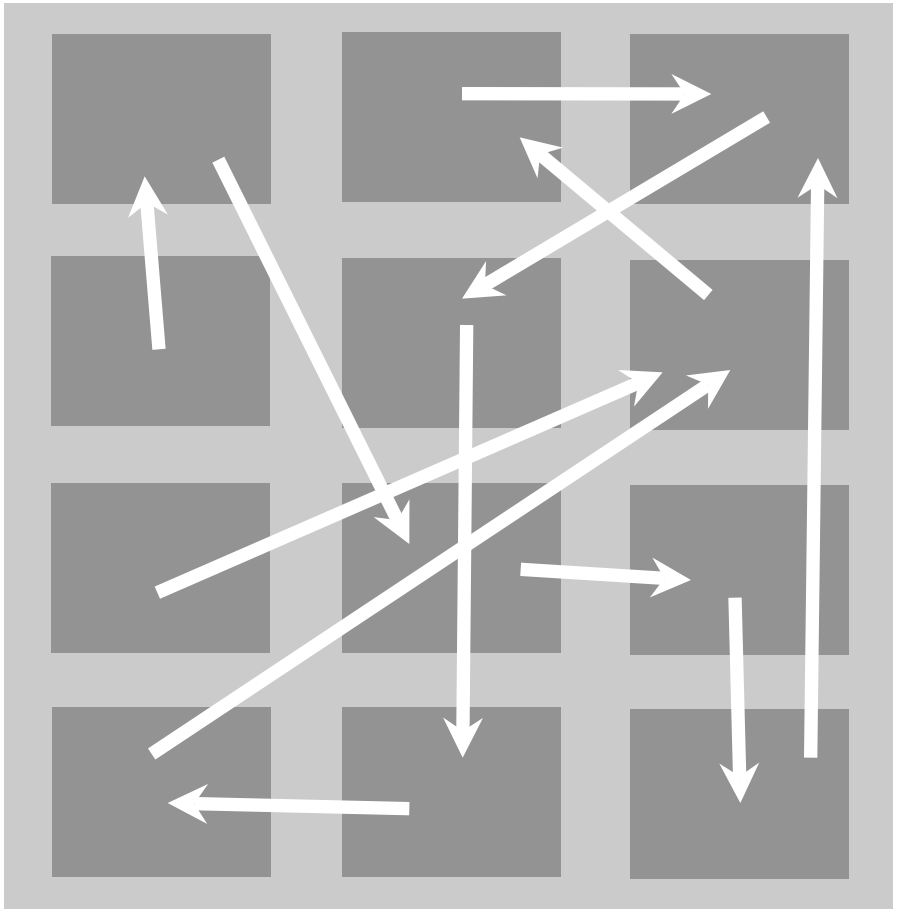
Centralized: every unit of time, each node sends a message to the master computer.

- **Drawback:** does not scale up well due to the communication congestion at the master computer.
  - May be suitable for medium size configurations, e.g., with thousands of nodes, but it is unlikely to be suitable for configurations with millions of nodes.

Hierarchical tree:

- Each node sends its information to its parent node until all the information arrives at the master computer in O(*log* # of nodes) units of time.
- Sensitive to node failures, *log* delay.

Randomized Gossip



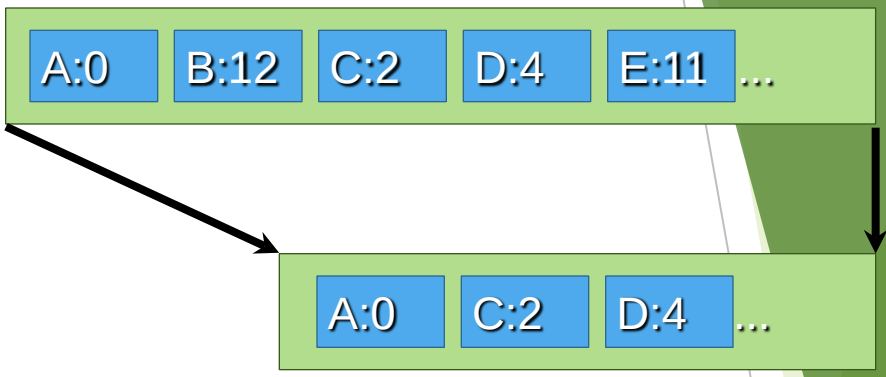
Distributed Bulletin Board

- Each node keeps vector with per-node info (own + info received from others)
- Once per time step, each node sends to 1 other randomly selected node a subset of its own vector entries (called “window”)
- Node merges received window entries into local vector (if newer)

MOSIX: Gossip Algorithm

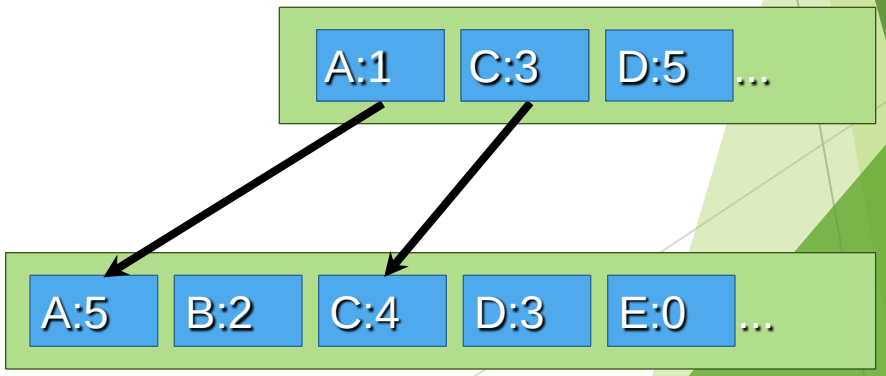
Each time unit:

- Update local info
- Find all vector entries up to age T (called a window)
- Send window to 1 randomly selected node



Upon receiving a window:

- Update the received entries' age (+1 for transfer)
- Update entries in local vector where newer information has been received



A two layer gossip algorithm

Compute nodes are divided into colonies.

- Based on some optimization criteria, e.g. network proximity.
- Colony nodes exchange local information for performing localized management decisions, such as load-balancing.
- Any client application in need of up-to-date information about the state of the resources in its colony can directly obtain it locally.

Master computer

- Collects information from a few nodes in each colony about all its nodes.
- Provide it to client process, e.g. scheduler.

Outcome: distributed bulletin board

Algorithm Trade-offs

- **Colony size**
  - Fresh limited information VS older wider one
- **Colony topology**
  - Spread VS focused node data
- **Dissemination rate**
  - Fresh information VS network overhead
- **Bigger threshold value**
  - Results in bigger window size
  - Lower average age VS more data being sent

Results – Average Information in Cluster (WIP)

Table 2: Average age of the whole vector.

Colony nodes	Method	Circulating among colony nodes: windows with T not exceeding age										Whole vector
		1	2	3	4	5	6	7	8	9	10	
128	Approx.	47.71	19.15	9.20	6.00	5.12	4.93	4.89	4.89	4.89	4.89	4.89
	Simul.	47.19	18.87	9.19	6.04	5.22	4.97	4.94	4.92	4.93	4.95	
	Emul.	45.71	18.78	9.19	5.96	5.13	4.92	4.89	4.89	4.88	4.88	
256	Approx.	94.80	36.49	15.63	8.49	6.28	5.70	5.59	5.57	5.57	5.57	5.57
	Simul.	94.17	36.33	15.48	8.57	6.39	5.77	5.64	5.63	5.63	5.62	
512	Approx.	188.98	71.15	28.41	13.27	8.20	6.70	6.33	6.26	6.25	6.25	6.25
	Simul.	187.97	71.01	28.38	13.34	8.32	6.81	6.41	6.34	6.32	6.32	
1K	Approx.	377.34	140.44	53.92	22.69	11.76	8.21	7.21	6.99	6.95	6.94	6.94
	Simul.	372.98	139.76	53.87	22.73	11.94	8.33	7.30	7.06	7.02	7.01	
2K	Approx.	754.05	279.03	104.91	41.47	18.73	10.90	8.44	7.79	7.66	7.63	7.63
	Simul.	710.19	267.82	104.87	41.58	18.96	11.08	8.58	7.89	7.73	7.71	
4K	Approx.	1507.47	556.20	206.87	78.99	32.56	16.06	10.50	8.83	8.42	8.34	8.32
	Simul.	1314.73	479.96	203.96	79.10	32.76	16.23	10.66	8.95	8.51	8.42	
8K	Approx.	3014.30	1110.53	410.80	154.02	60.17	26.26	14.34	10.44	9.32	9.07	9.01
	Simul.		798.97	390.69	153.80	60.36	26.48	14.54	10.59	9.43		
1M	Approx.	385,750	141,911	52,208	19,209	7,070	2,605	963	360	138	58	13.86
1G	Approx.	395M	145M	53M	19M	7M	2M	979K	360K	132K	48K	20.79

Results – Average Information in Master (WIP)

Table 3: Average age of the master computer entries using Algorithm 1, $k = 1$.

Colony nodes	Method	Circulating among colony nodes, windows with T not exceeding age: 2	3	4	5	6	7	8	9	10	Whole vector
128	Approx.	5.59	4.46	4.04	3.93	3.92	3.91	3.91	3.91	3.91	3.91
	Simul.	5.35	4.09	3.77	3.70	3.68	3.68	3.67	3.67	3.70	
	Emul.	5.73	4.74	4.26	4.33	4.16	4.06	4.09	4.15	4.15	
256	Approx.	7.75	5.80	4.94	4.65	4.59	4.58	4.58	4.58	4.58	4.58
	Simul.	7.53	5.47	4.69	4.42	4.38	4.36	4.37	4.33	4.40	
512	Approx.	10.79	7.67	6.13	5.49	5.30	5.26	5.26	5.26	5.26	5.26
	Simul.	10.33	7.26	5.83	5.19	5.04	5.00	4.99	5.01	5.03	
1K	Approx.	15.11	10.29	7.75	6.55	6.09	5.96	5.94	5.94	5.94	5.94
	Simul.	14.45	10.01	7.45	6.32	5.85	5.71	5.71	5.74	5.69	
2K	Approx.	21.21	14.00	10.01	7.96	7.04	6.72	6.64	6.63	6.63	6.63
	Simul.	20.90	13.72	9.78	7.73	6.78	6.52	6.46	6.42	6.38	
4K	Approx.	29.85	19.24	13.19	9.91	8.27	7.58	7.37	7.33	7.32	7.32
	Simul.	29.97	19.10	12.78	9.61	8.01	7.29	7.12	7.10	7.11	
8K	Approx.	42.07	26.65	17.69	12.64	9.96	8.68	8.18	8.04	8.02	8.01
	Simul.										
1M	Approx.	472.47	287.70	176.02	108.68	68.23	44.09	29.84	21.59	16.98	12.86
1G	Approx.	1331.16	1328.35	1320.12	1297.13	1233.92	1068.77	752.79	463.56	285.05	19.79