Weighted Available Space Selection

Fixing write pool selection

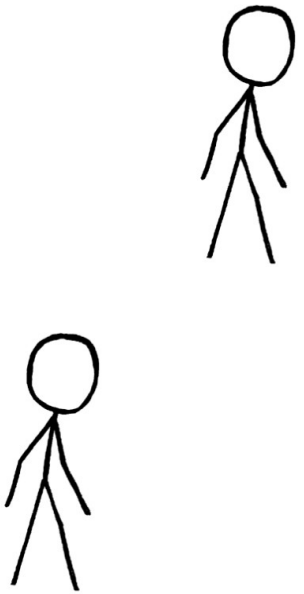
Mattias Wadenstein

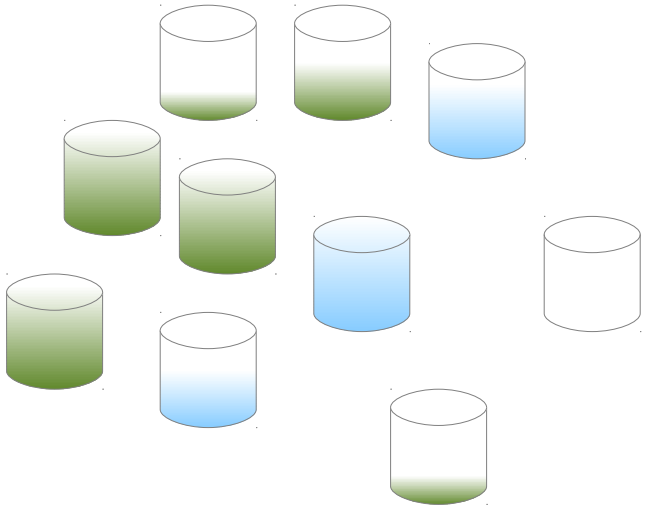
maswan@hpc2n.umu.se

Gerd Behrmann

behrmann@nordu.net

Static pool selection





- By VO
- By user
- By type of data
- By path

- By protocol
- By network
- Static routing table maps transfers to a set of pools

Dynamic pool selection

- By size
- By free space
- By age
- By load

- By queue length
- By queue capacity

Outline

- Both static and dynamic pool selection is pluggable in dCache.
- This talk is about dynamic pool selection of write pools.
- Read pool selection is interesting too, but usually constrained by only having one or few copies of a file.

- What are the problems with the traditional pool selection in dCache?
- What can be done to solve them?

Desirable properties

- Must balance load
- Avoids temporal clumping of files
- Is unaffected by disk partioning
- Work for heterogeneous pools
  - Pool size
  - Throughput
  - Spindles
  - Age
- Backs off in case of overload
- Takes age into account for garbage collection
- Avoids oscilations
- Is distributed
- Maintains performance over time (no phase transitions)

Issues with pool selection

- Difficult to tune

$$cost(p, fileSize) = cpuCostFactor \cdot \frac{\sum_{q \in queues(p)} \frac{movers(q)}{limit(q)}}{|queues(p)|} + spaceCostFactor \cdot \frac{3 \cdot fileSize}{free(p)}$$

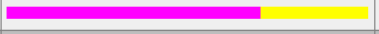
Issues with pool selection

- Difficult to tune

$$cost(p, fileSize) = cpuCostFactor \cdot \frac{\sum_{q \in queues(p)} \frac{movers(q)}{limit(q)}}{|queues(p)|} + spaceCostFactor \cdot \frac{3 \cdot fileSize}{free(p)}$$

- Write clumping

Disk Space Usage

CellName	DomainName	Total Space/MiB	Free Space/MiB	Precious Space/MiB	Layout (precious/used/free)	SpaceCost (50MB)
pool_0	pool	1607	1116	15		0,134
pool_1	pool	1024	305	0		0,492
pool_2	pool	2482	1116	48		0,134

Issues with pool selection

- Difficult to tune

$$cost(p, fileSize) = cpuCostFactor \cdot \frac{\sum_{q \in queues(p)} \frac{movers(q)}{limit(q)}}{|queues(p)|} + spaceCostFactor \cdot \frac{3 \cdot fileSize}{free(p)}$$

- Write clumping
- Cost linked to concurrency limits
- Hot reads push away writes

Pool Request Queues

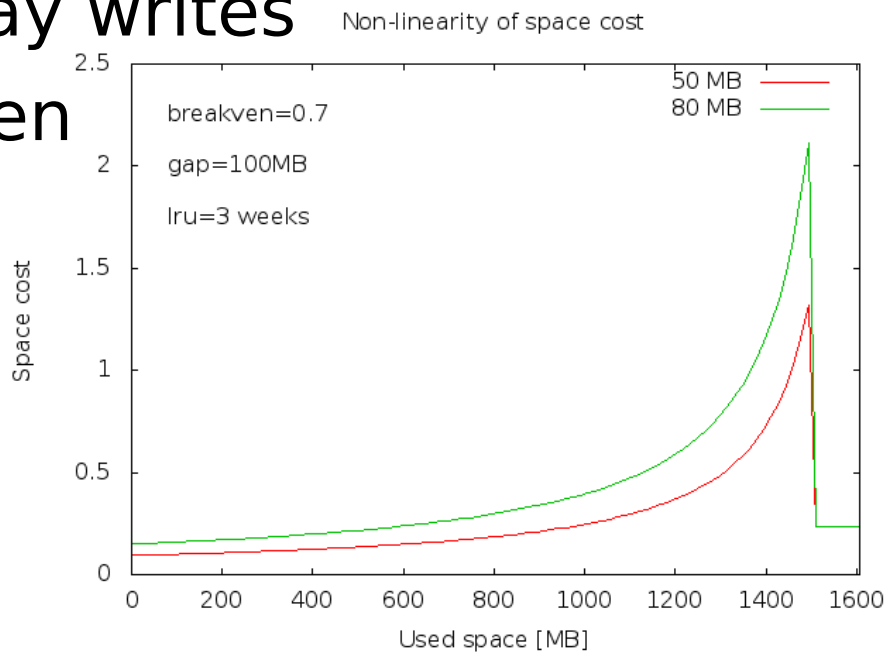
CellName	CpuCost	Restores			Stores			P2P-Server			P2P-Client			regular		
		Active	Max	Queued	Active	Max	Queued	Active	Max	Queued	Active	Max	Queued	Active	Max	Queued
Total		0	1	0	1	1	74	0	21	0	0	30	0	0	300	0
pool_0	3.60	0	1	0	1	1	17	0	1	0	0	10	0	0	100	0
pool_1	0.00	0	0	0	0	0	0	0	10	0	0	10	0	0	100	0
pool_2	0.00	0	0	0	0	0	57	0	10	0	0	10	0	0	100	0
Total		0	1	0	1	1	74	0	21	0	0	30	0	0	300	0
CellName	CpuCost	Active	Max	Queued	Active	Max	Queued	Active	Max	Queued	Active	Max	Queued	Active	Max	Queued
		Restores			Stores			P2P-Server			P2P-Client			regular		

Issues with pool selection

- Difficult to tune

$$cost(p, fileSize) = cpuCostFactor \cdot \frac{\sum_{q \in queues(p)} \frac{movers(q)}{limit(q)}}{|queues(p)|} + spaceCostFactor \cdot \frac{3 \cdot fileSize}{free(p)}$$

- Write clumping
- Cost linked to concurrency limits
- Hot reads push away writes
- Non-linear shift when pool becomes full



Issues with pool selection

- Difficult to tune

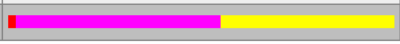
$$cost(p, fileSize) = cpuCostFactor \cdot \frac{\sum_{q \in queues(p)} \frac{movers(q)}{limit(q)}}{|queues(p)|} + spaceCostFactor \cdot \frac{3 \cdot fileSize}{free(p)}$$

- Write clumping
- Cost linked to concurrency limits
- Hot reads push away writes
- Non-linear shift when pool becomes full
- Tends to operate in modes
- Requires up-to-date information
- Stability issues

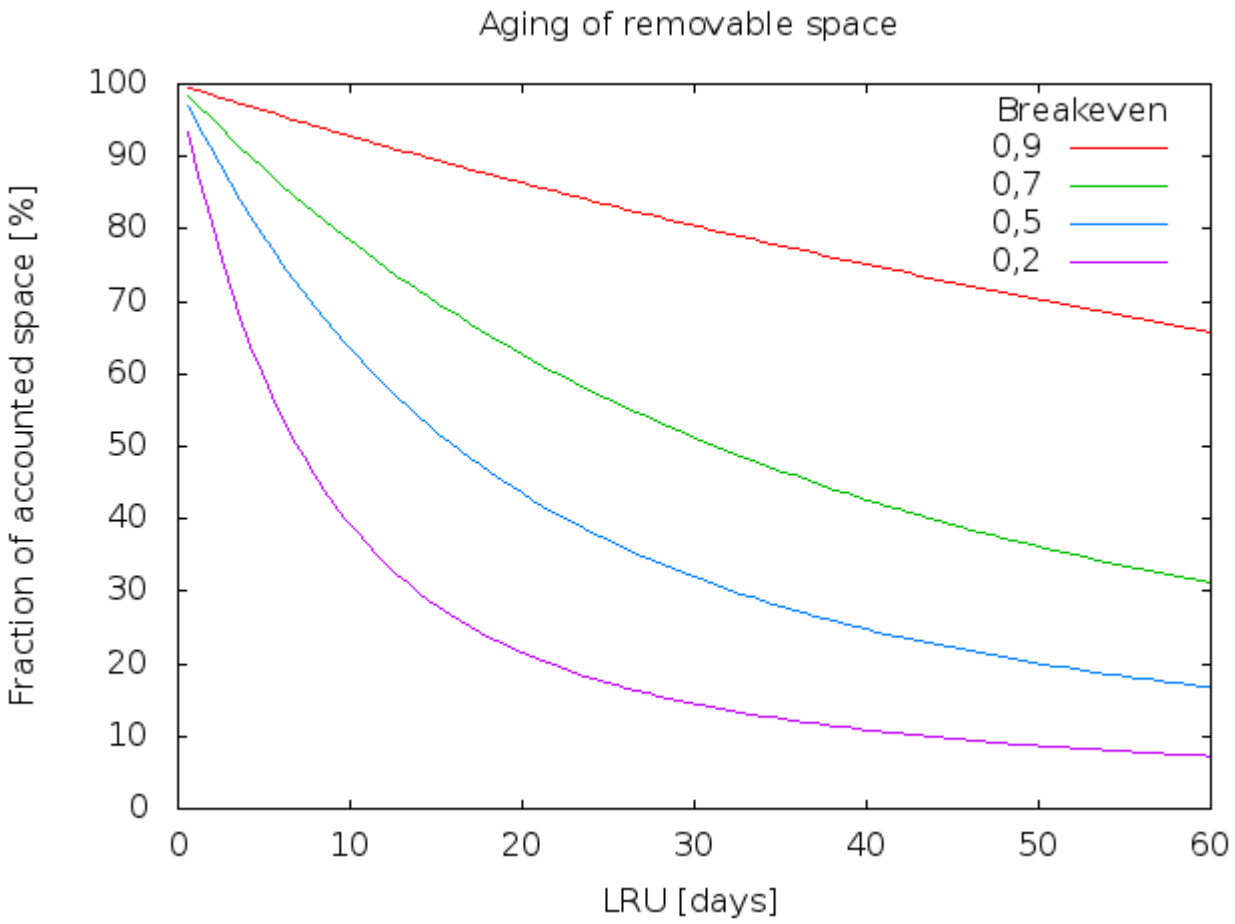
WASS

- Weighted Available Space Selection

Disk Space Usage

CellName	DomainName	Total Space/MiB	Free Space/MiB	Precious Space/MiB	Layout (precious/used/free)	Available Space (MiB)	P
pool_0	pool	1607	1116	15			44%
pool_1	pool	1024	305	0			12%
pool_2	pool	2482	1116	48			44%

Removable space



Removable space

- Weighted Available Space Selection
  - Exponential decay of removable space

Disk Space Usage

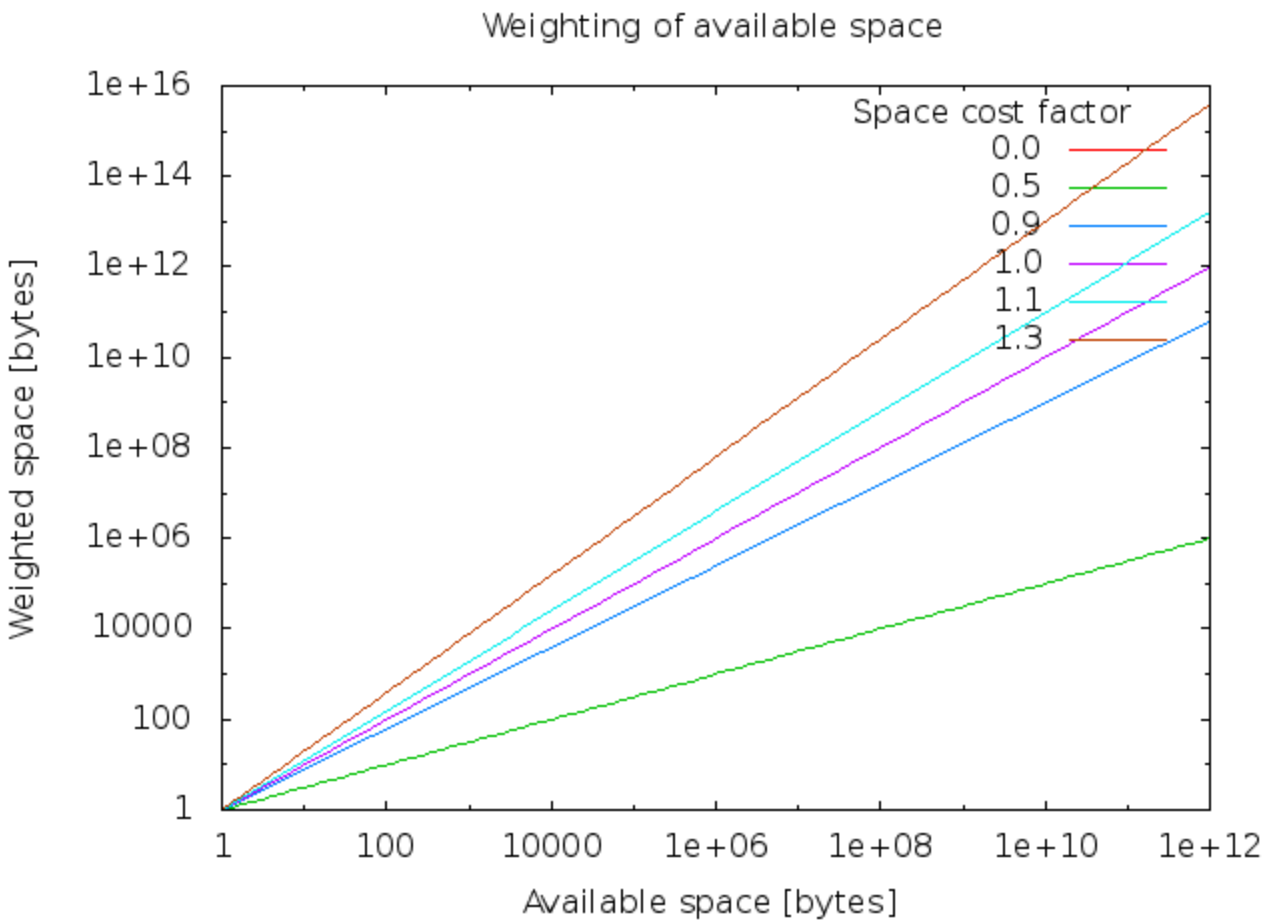
CellName	DomainName	Total Space/MiB	Free Space/MiB	Precious Space/MiB	Layout (precious/used/free)	Available Space (MiB)	P
pool_0	pool	1607	1116	15			40%
pool_1	pool	1024	305	0			20%
pool_2	pool	2482	1116	48			40%

Space cost factor

$$weighted = available^{scf}$$

Space cost factor

$$weighted = available^{scf}$$



Space cost factor

- Weighted Available Space Selection
  - Exponential decay of removable space

0.0

CellName	DomainName	Total Space/MiB	Free Space/MiB	Precious Space/MiB	Layout (precious/used/free)	Weighted Available Space (MiB)	P
pool_0	pool	1607	1116	15			33.3%
pool_1	pool	1024	305	0			33.3%
pool_2	pool	2482	1116	48			33.3%

0.5

CellName	DomainName	Total Space/MiB	Free Space/MiB	Precious Space/MiB	Layout (precious/used/free)	Weighted Available Space (MiB)	P
pool_0	pool	1607	1116	15			36.8%
pool_1	pool	1024	305	0			26.3%
pool_2	pool	2482	1116	48			36.8%

1.0

CellName	DomainName	Total Space/MiB	Free Space/MiB	Precious Space/MiB	Layout (precious/used/free)	Available Space (MiB)	P
pool_0	pool	1607	1116	15			40%
pool_1	pool	1024	305	0			20%
pool_2	pool	2482	1116	48			40%

1.5

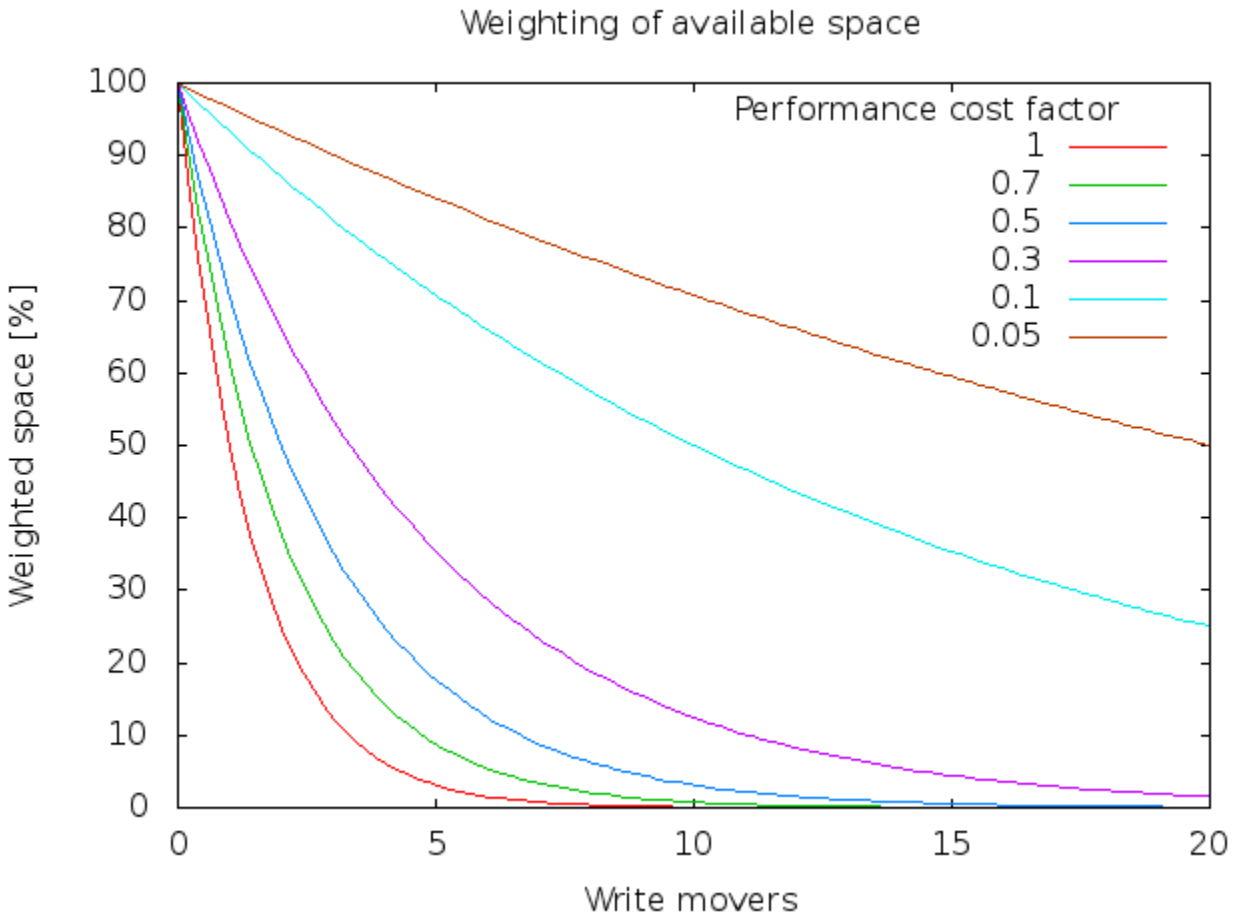
CellName	DomainName	Total Space/MiB	Free Space/MiB	Precious Space/MiB	Layout (precious/used/free)	Weighted Available Space (MiB)	P
pool_0	pool	1607	1116	15			42.4%
pool_1	pool	1024	305	0			15.3%
pool_2	pool	2482	1116	48			42.4%

Performance cost factor

$$weighted = \frac{available^{scf}}{2^{mcf \cdot ccf \cdot writers}}$$

Performance cost factor

$$weighted = \frac{available^{scf}}{2^{mcf \cdot ccf \cdot writers}}$$



Performance cost factor

0.5

Pool 1			Pool 2			Pool 3		
Writers	Weight	P	Writers	Weight	P	Writers	Weight	P
0	1.00	33.3%	0	1.00	33.3%	0	1.00	33.3%
5	0.176	33.3%	5	0.176	33.3%	5	0.176	33.3%
0	1.0	82.9%	5	0.176	14.6%	10	0.031	2.6%
100	8.8^{-16}	82.9%	105	1.6^{-16}	14.6%	110	2.7^{-17}	2.6%

0.2

Pool 1			Pool 2			Pool 3		
Writers	Weight	P	Writers	Weight	P	Writers	Weight	P
0	1.00	33.3%	0	1.00	33.3%	0	1.00	33.3%
5	0.50	33.3%	5	0.50	33.3%	5	0.50	33.3%
0	1.00	57.1%	5	0.50	28.6%	10	0.25	14.3%
100	9.5^{-7}	57.1%	105	4.7^{-7}	28.6%	110	2.4^{-7}	14.3%

Parameters

- Break even
- Performance cost factor
  - Mover cost factor
  - CPU cost factor
- Space cost factor

Summary

- Default in dCache 2.2 for new systems
- Has been used at NDGF for 8 months (and has worked well)
- Should work well with default values
- Accuracy of removable space aging can be improved