Hoarding for a Hierarchical Storage Architecture.

Christopher LaRosa ’03

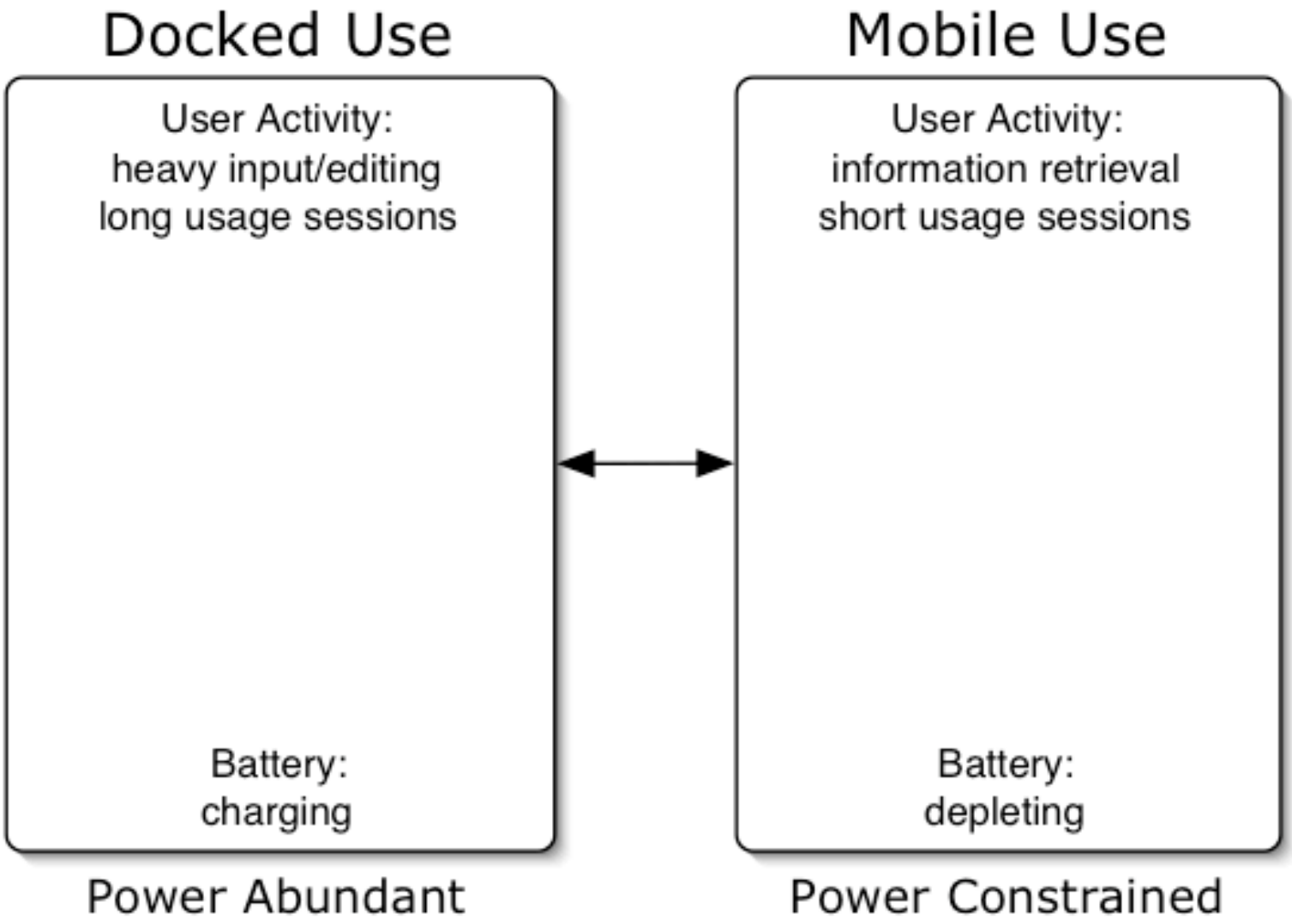
Computer Science Department
Hamilton College
May 12, 2003

Hardware Similarity/Disparity

3 y/o Laptop Computer — **Current Handheld Computer**

- 400 Mhz. RISC
- 20 Gbyte disk
- 64 Mbyte RAM
- 1024 x 768 pixel display
- fully-sized keyboard

- 400 Mhz. CISC
- 48 Mbyte flash memory
- 64 Mbyte RAM
- 320 x 240 pixel display
- awkard input


Docked Use
User Activity:
heavy input/editing
long usage sessions
Battery:
charging
Power Abundant
Mobile Use
User Activity:
information retrieval
short usage sessions
Battery:
depleting
Power Constrained

Previous Research

Project	**Focus – Approach**
CODA (Sat02)	Provide data to disconnected clients in laptop/server environment – Aggressively hoard files, LRU based and user directed file hoarding.

Previous Research

Project	**Focus – Approach**
CODA (Sat02)	Provide data to disconnected clients in laptop/server environment – Aggressively hoard files, LRU based and user directed file hoarding.
SEER (Kue97)	Data availability during disconnect – Develop projects by tracking the interval between different file accesses. Hoard frequently accessed files and their closely related files.

Previous Research

Project	**Focus – Approach**
CODA (Sat02)	Provide data to disconnected clients in laptop/server environment – Aggressively hoard files, LRU based and user directed file hoarding.
SEER (Kue97)	Data availability during disconnect – Develop projects by tracking the interval between different file accesses. Hoard frequently accessed files and their closely related files.
Tree-Based (Tai95)	Data availability during disconnect – Develop file trace trees that represent typical file use for each application. Hoard recently used applications' file trace trees.

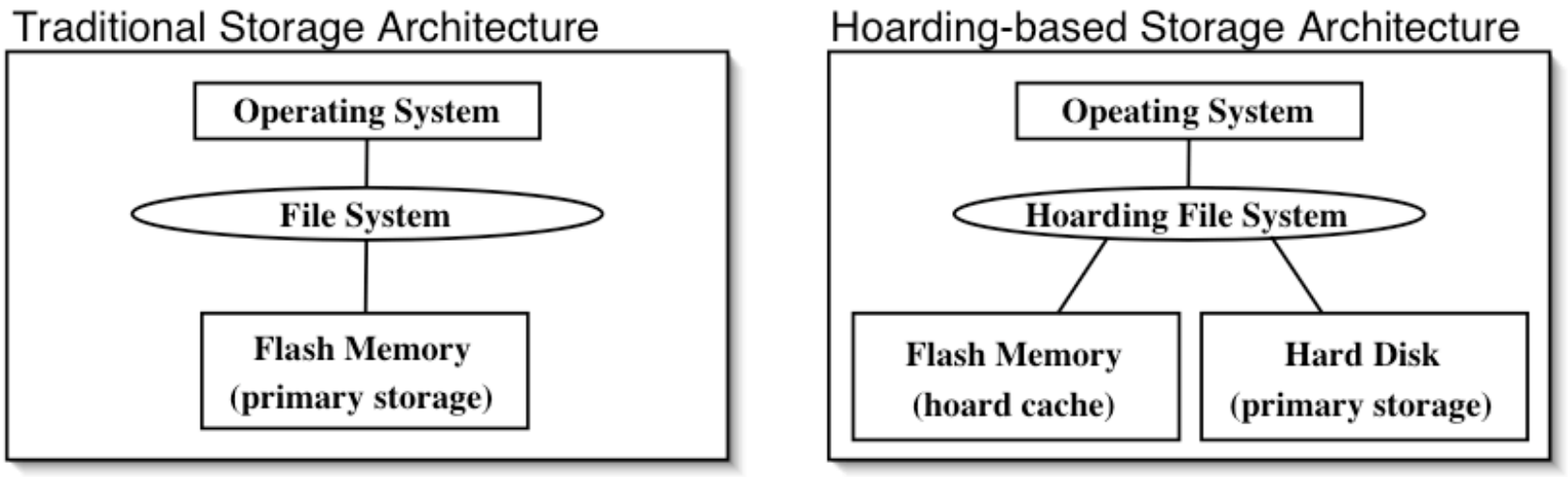
Previous Research

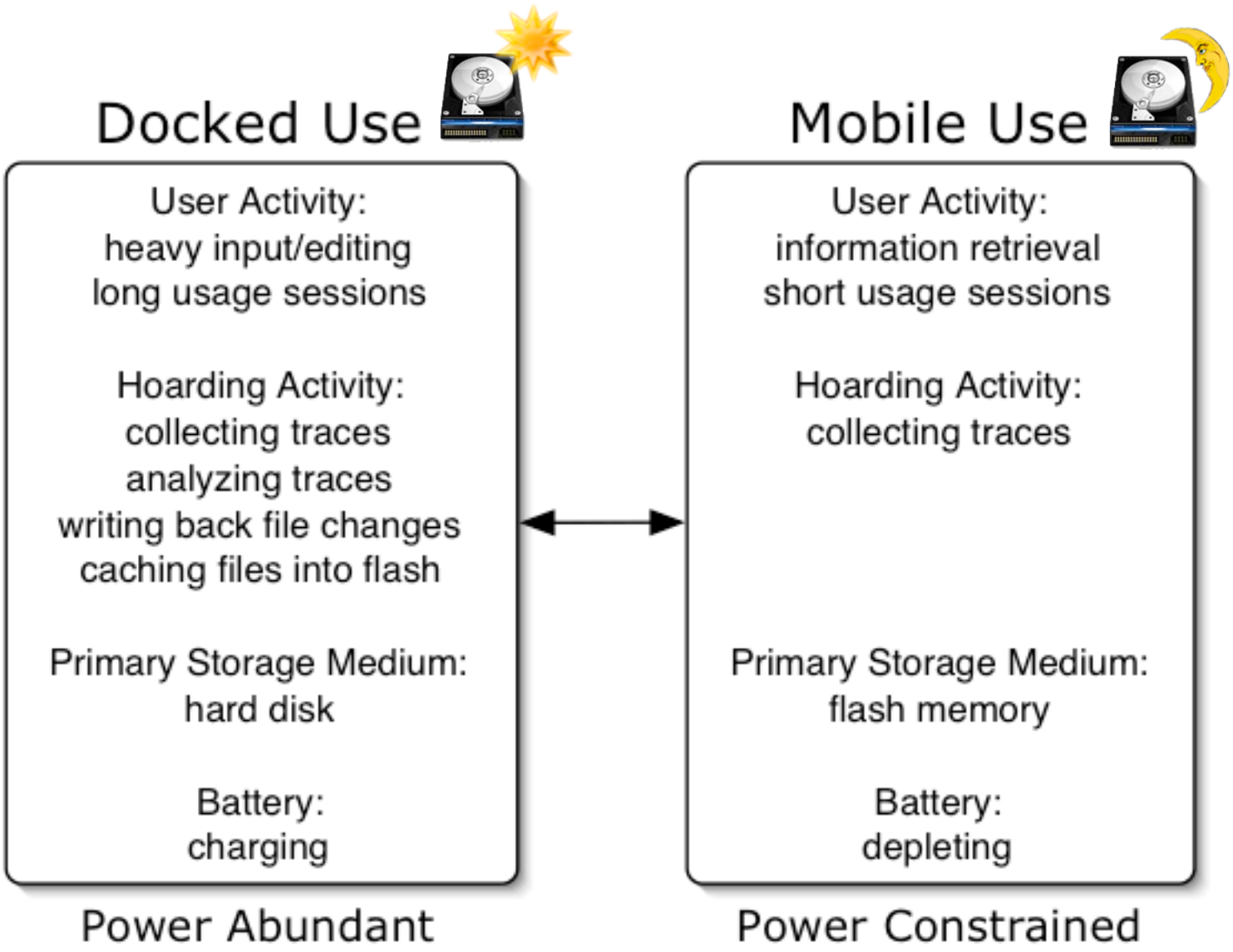
Project	**Focus – Approach**
CODA (Sat02)	Provide data to disconnected clients in laptop/server environment – Aggressively hoard files, LRU based and user directed file hoarding.
SEER (Kue97)	Data availability during disconnect – Develop projects by tracking the interval between different file accesses. Hoard frequently accessed files and their closely related files.
Tree-Based (Tai95)	Data availability during disconnect – Develop file trace trees that represent typical file use for each application. Hoard recently used applications' file trace trees.
FBR (Rob90)	Improving hit ratios for file caches – Account for the importance of long-term repetition in access for file caches. Roughly 30% efficiency improve. over LRU for file caches.

Previous Research

Project	**Focus – Approach**
CODA (Sat02)	Provide data to disconnected clients in laptop/server environment – Aggressively hoard files, LRU based and user directed file hoarding.
SEER (Kue97)	Data availability during disconnect – Develop projects by tracking the interval between different file accesses. Hoard frequently accessed files and their closely related files.
Tree-Based (Tai95)	Data availability during disconnect – Develop file trace trees that represent typical file use for each application. Hoard recently used applications' file trace trees.
FBR (Rob90)	Improving hit ratios for file caches – Account for the importance of long-term repetition in access for file caches. Roughly 30% efficiency improve. over LRU for file caches.
Process Offload (Li01)	Improve efficiency of costly computation – Offload processor intensive tasks to energy abundant servers, focus on developing heuristics to calculate efficient division.

Traditional *vs* Hoarding-based Storage Architectures




Docked Use
User Activity:
heavy input/editing
long usage sessions
Hoarding Activity:
collecting traces
analyzing traces
writing back file changes
caching files into flash
Primary Storage Medium:
hard disk
Battery:
charging
Power Abundant
Mobile Use
User Activity:
information retrieval
short usage sessions
Hoarding Activity:
collecting traces
Primary Storage Medium:
flash memory
Battery:
depleting
Power Constrained

Footprint Comparison

Toshiba HDD1262 Footprint
Zaurus SL-5600 Footprint

Modeling Power Cost Difference

- cost difference between mediums
- incidental costs for mediums
  - flash memory
    - none
  - hard disk
    - spin up (disk cost + overhead cost*)
    - idle spin time during inactivity threshold

$$C_{diff} = \left[\sum_{i}^{n} (D_i \cdot F_i) + S_t \left[O_c + S_c \right] \cdot S(n) + D_{idle} * I(n,t) \right]$$

Modeling Power Cost Difference

- cost difference between mediums
- incidental costs for mediums
  - flash memory
    - none
  - hard disk
    - spin up (disk cost + overhead cost*)
    - idle spin time during inactivity threshold

$$C_{diff} = q \square\left[(D_{ave} \square F_{ave})\right] + S_t\left[O_c + S_c\right]S(n) + D_{idle} * I(n,t)$$

Modeling Battery Runtime (Life)

$$R_{orig} = \frac{batterycapacity(watthours)}{averagedraw(watts)}$$

$$R_{hhs} = \frac{batterycapacity(watthours)}{averagedraw(watts)+C_{diff}}$$

$$R_{\%} = \frac{R_{hhsa}}{R_{orig}} = \frac{averagedraw(watts)}{averagedraw(watts)+C_{diff}}$$

Trace Data Using LTT

trace name	total files	total data size	average file size	ave. access interval
15 minute a	504	111.6 MB	226.7 KB	1.8
15 minute b	502	114.2 MB	232.9 KB	1.8
2 minute a	182	87.7 MB	493.4 KB	.7
2 minute b	43	11.1 MB	264.3 KB	2.8
2 minute c	45	6.9 MB	157.0 KB	2.7

Fig 5.1 – Trace Statistics

Overall average file size ≈ 250 KB

Cache Performance Using Frequency Based Hoarding

trace	250 file cache ≈ 64 MB Cache				400 file cache ≈ 96 MB			
	hits	misses	hit rate	interval	hits	misses	hit rate	interval
2 min. a	61	121	.34	.99	110	72	.60	1.66
2 min. b	22	21	.51	5.74	28	15	.65	8.00
2 min. c	21	24	.47	5.00	26	19	.57	6.31

Fig 5.2 – Simulation results with *15 minute a* as Hoard List Generator input

Cache Performance Using Frequency Based Hoarding

trace	250 file cache ≈ 64 MB Cache				400 file cache ≈ 96 MB			
	hits	misses	hit rate	interval	hits	misses	hit rate	interval
2 min. a	61	121	.34	.99	110	72	.60	1.66
2 min. b	22	21	.51	5.74	28	15	.65	8.00
2 min. c	21	24	.47	5.00	26	19	.57	6.31

Fig 5.2 – Simulation results with *15 minute a* as Hoard List Generator input

trace name	250 file cache ≈ 64 MB Cache				400 file cache ≈ 96 MB			
	hits	misses	hit rate	interval	hits	misses	hit rate	interval
2 min. b	29	14	.67	8.57	34	9	.79	13.30
2 min. c	30	15	.67	8.00	36	9	.80	13.30

Fig 5.3 – Simulation results with multiple traces as Hoard List Generator input.

Cache Performance Using Frequency Based Hoarding

trace	250 file cache ≈ 64 MB Cache				400 file cache ≈ 96 MB			
	hits	misses	hit rate	interval	hits	misses	hit rate	interval
2 min. a	61	121	.34	.99	110	72	.60	1.66
2 min. b	22	21	.51	5.74	28	15	.65	8.00
2 min. c	21	24	.47	5.00	26	19	.57	6.31

Fig 5.2 – Simulation results with *15 minute a* as Hoard List Generator input

trace name	250 file cache ≈ 64 MB Cache				400 file cache ≈ 96 MB			
	hits	misses	hit rate	interval	hits	misses	hit rate	interval
2 min. b	29	14	.67	8.57	34	9	.79	13.30
2 min. c	30	15	.67	8.00	36	9	.80	13.30

Fig 5.3 – Simulation results with multiple traces as Hoard List Generator input.

trace name	250 file cache ≈ 64 MB Cache				400 file cache ≈ 96 MB			
	hits	miss	hit rate	interval	hits	miss	hit rate	interval
2 min. b	29	8	.78	15	34	3	.91	40.00
2 min. c	30	15	.67	8.00	36	9	.80	13.30

Fig 5.4 – Simulation results with multiple traces as Hoard List Generator input and no Mozilla file cache.

Battery Life Impact

trace name	250 file cache ≈ 64 MB Cache, 5/10 second spin down threshold						
	miss	*I(n,t)* idle spin time	*S(t)* # spin ups	% runtime orig. - disruptive	% runtime orig - non-disruptive	% runtime continuous disk - disruptive	% runtime continuons disk - non-disruptive
2 min. b	8	22/42	4/4	.84/.82	.93/.91	.99/.96	1.10./1.07
2 min c	15	43/61	6/3	.77/.83	.89/.89	.91/.97	1.05/1.05

Battery Life Impact

trace name	250 file cache □ 64 MB Cache, 5/10 second spin down threshold						
	miss	*I(n,t)* idle spin time	*S(t)* # spin ups	% runtime orig. - disruptive	% runtime orig - non-disruptive	% runtime continuous disk - disruptive	% runtime continuons disk - non-disruptive
2 min. b	8	22/42	4/4	.84/.82	.93/.91	.99/.96	1.10./1.07
2 min c	15	43/61	6/3	.77/.83	.89/.89	.91/.97	1.05/1.05

trace name	400 file cache □ 96 MB Cache, 5/10 second spin down threshold						
	miss	*I(n,t)* idle spin time	*S(t)* # spin ups	% runtime orig - disruptive	% runtime orig. - non-disruptive	% runtime continuous disk - disruptive	% runtime continuons disk - non-disruptive
2 min. b	3	10/20	2/2	.91/.90	.97/.95	1.08/1.06	1.13/1.12
2 min c	9	27/42	3/3	.86/.85	.93/.92	1.02/1.00	1.10/1.08

Conclusions

- Copious historical trace data is imperative to hoardings success.
- Application settings can negatively affect hoarding success.
- Hoarding during an energy abundant docked state can greatly reduce the power cost associated with disk based mass storage.

Future Work

- Improve hoarding algorithm to near 100% hit rate.
- Investigate optimal idle spin threshold for handheld computers.
- Determine where utility gained from increasing trace data ceases to exist.
- Gather better, more abundant trace data reflecting usage in single device paradigm.

Hoarding for a Hierarchical Storage Architecture.

Christopher LaRosa ’03

Computer Science Department
Hamilton College
May 12, 2003