e-business

Striving for Optimal Journal Performance

BP03

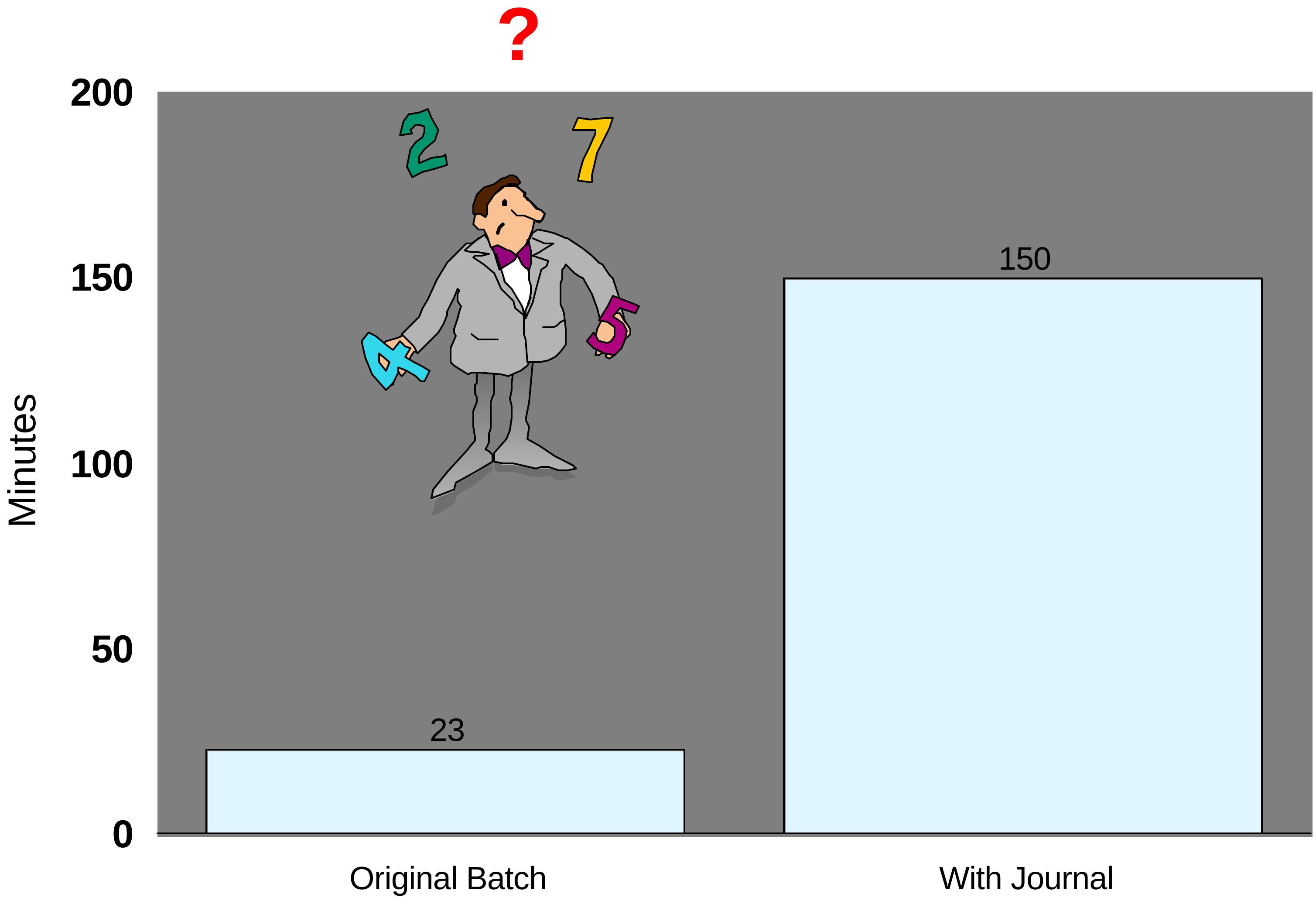
ITSO iSeries Technical Forum

Larry Youngren

Redbooks

International Technical Support Organization

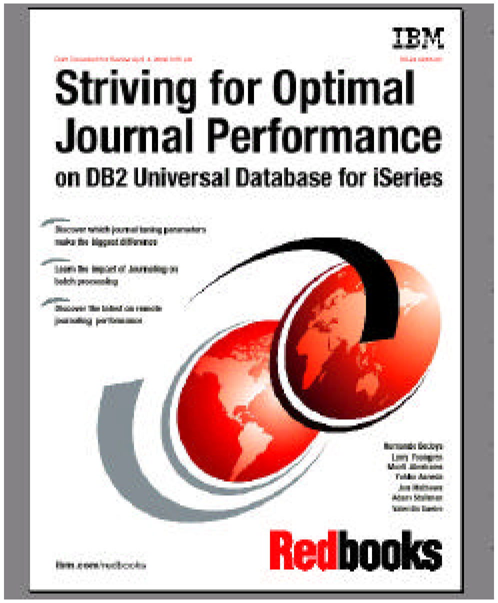
Tuning Batch Jobs (Case Study)



New Journal Performance Redbook

- We conducted journal performance studies
- The results are documented in a resulting **Redbook:**
  - **Striving for Optimal Journal Performance**
  - Order number
    - **SG24-6286-00.**
  - Web Site:
    - **http://www.redbooks.ibm.com**

Published: May of 2002



IBM

[illegible]

Striving for Optimal Journal Performance

on DB2 Universal Database for iSeries

Discover which journal tuning parameters make the biggest difference

Learn the impact of journaling on batch processing

Discover the latest on remote journaling performance

Hernando Bedoya
Larry Youngren
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]

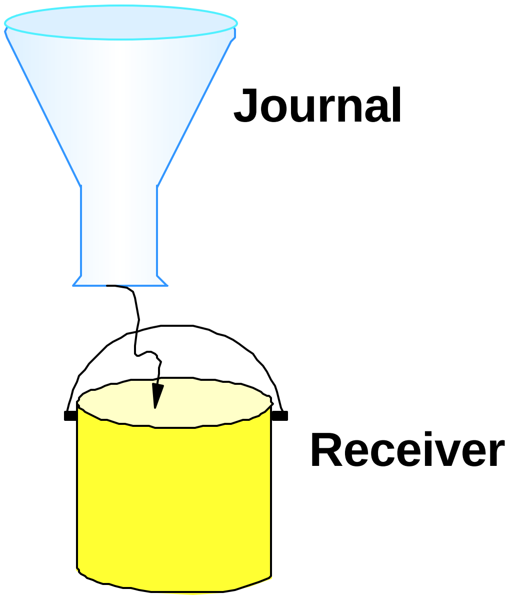
Redbooks

ibm.com/redbooks

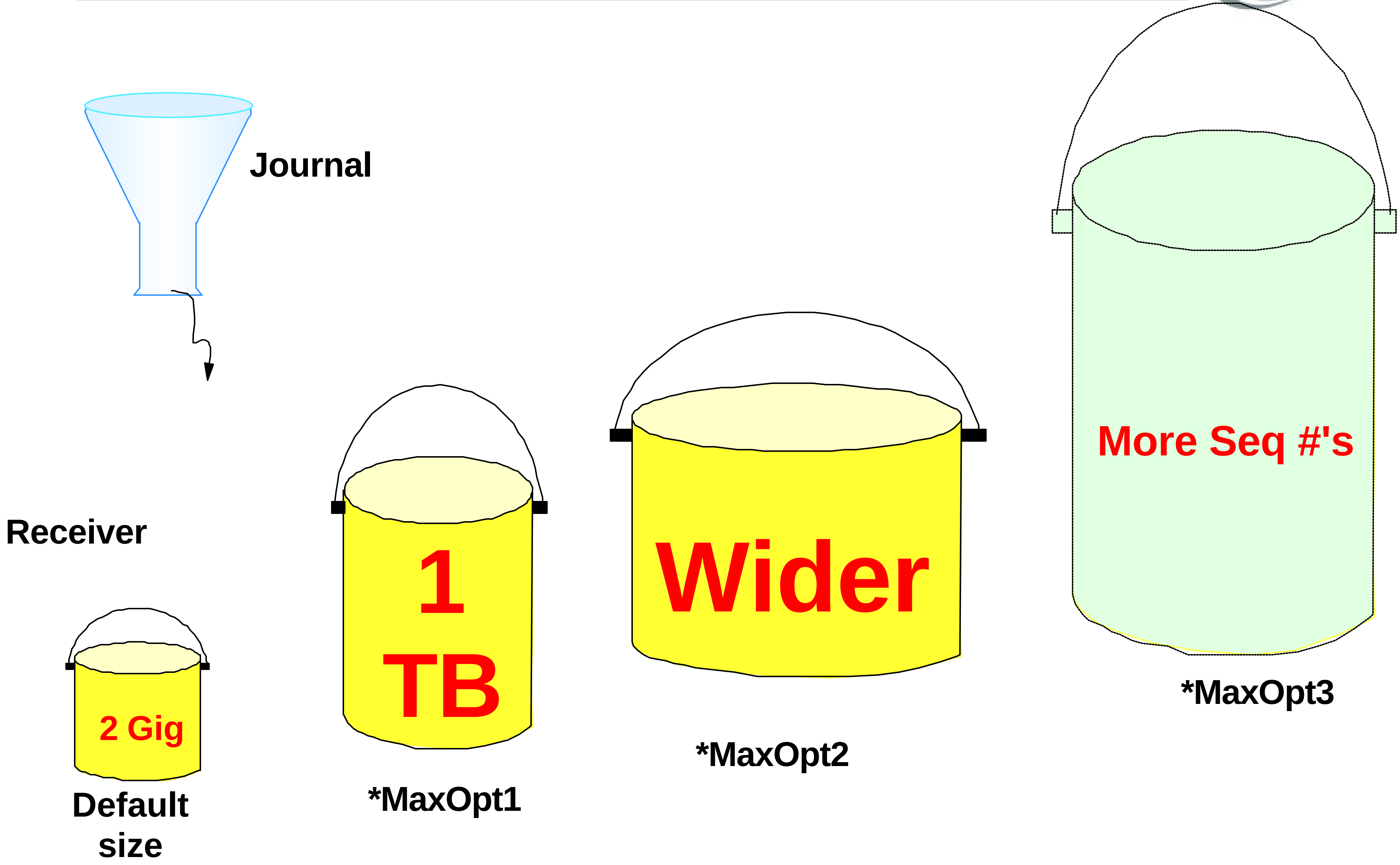
Three approaches

- **Hardware selection/configuration choices**
- **Journal tuning**
- **Application tuning { Make 'em Journal Friendly }**

The Objects



Super-size them !



Journal

Receiver

2 Gig

Default size

1 TB

*MaxOpt1

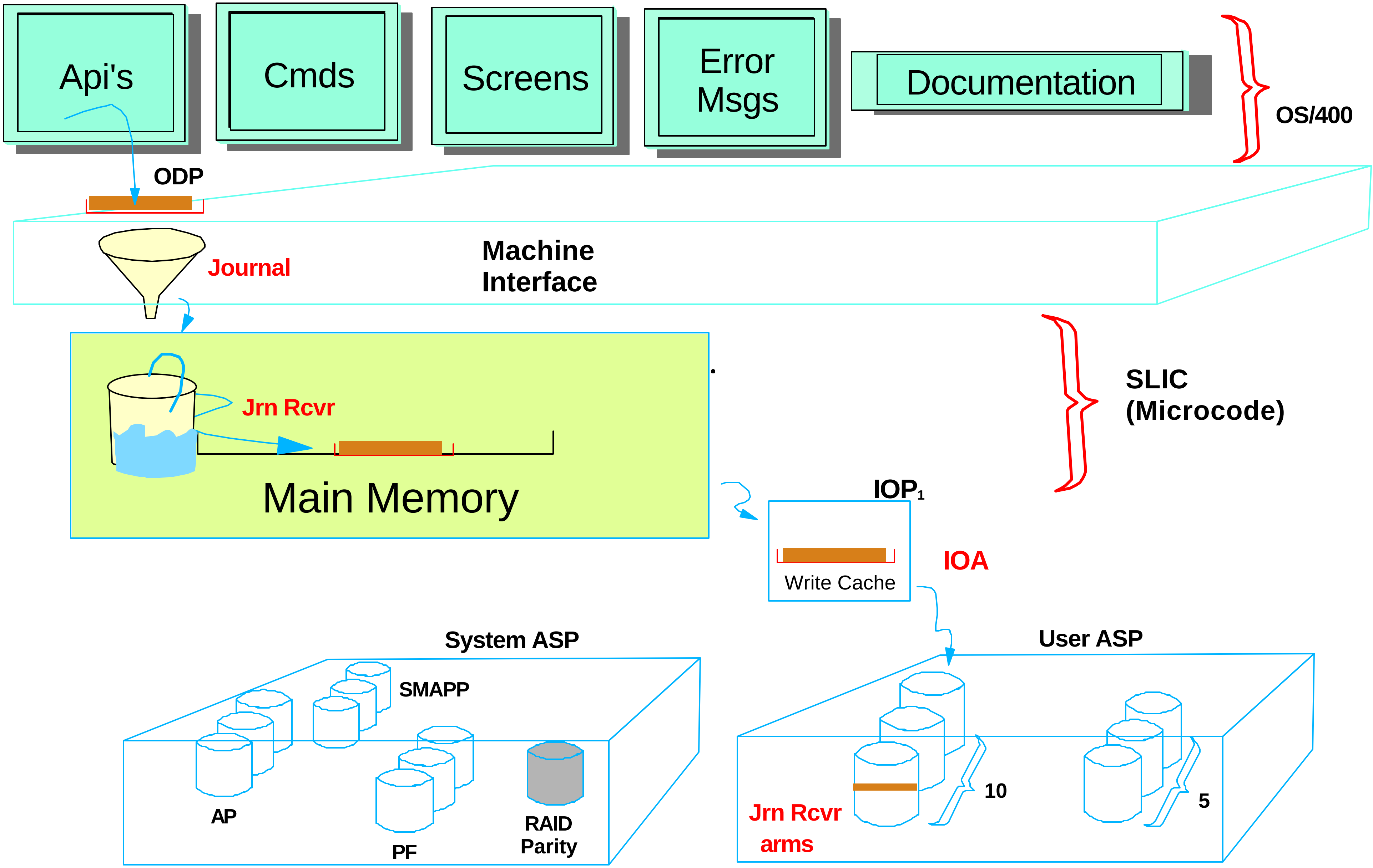
Wider

*MaxOpt2

More Seq #'s

*MaxOpt3

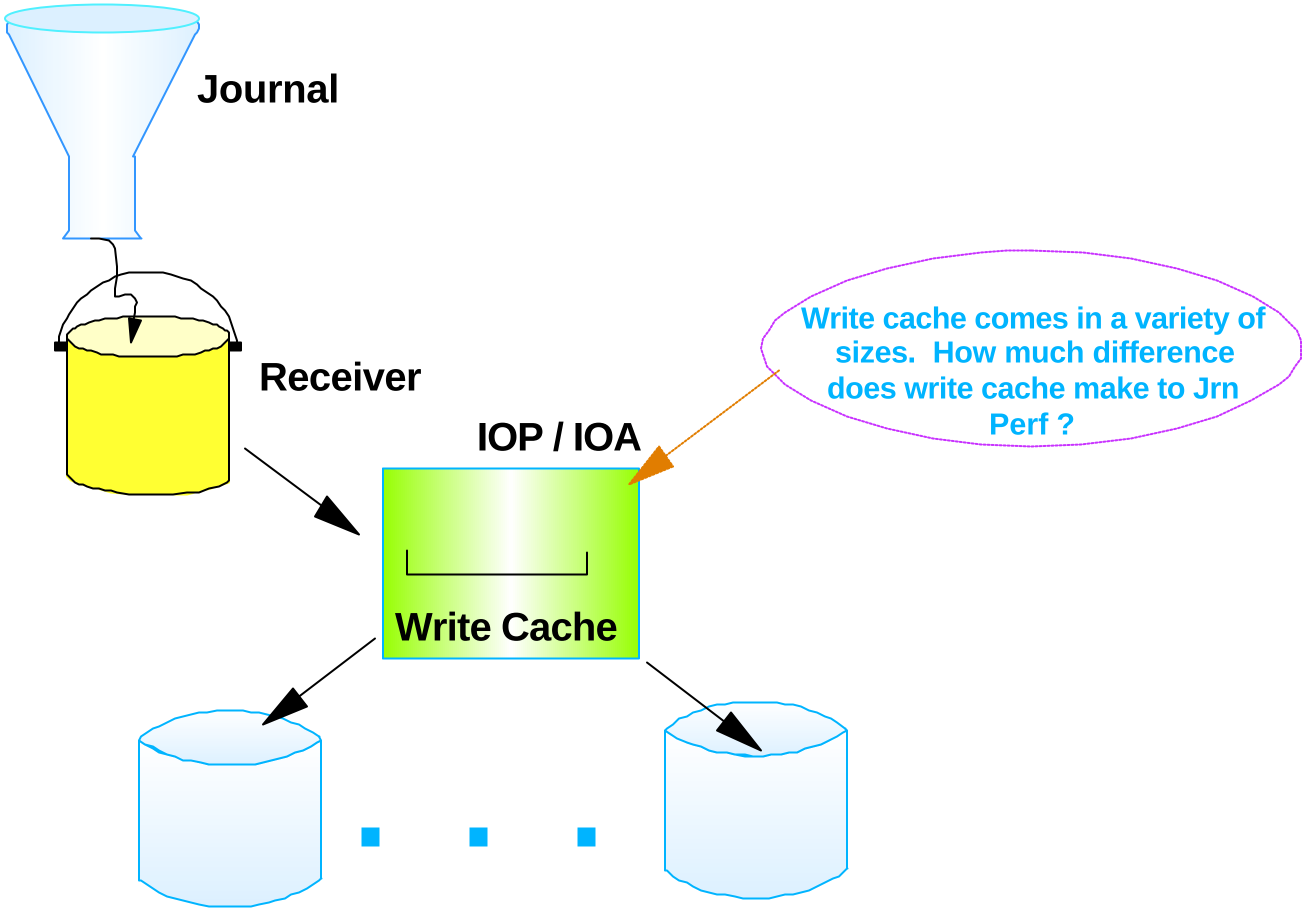
The Journal environment



Hardware & Configurations

Sizing the IOA write cache
(Does it matter ?)

How much IOA write cache do I need ?



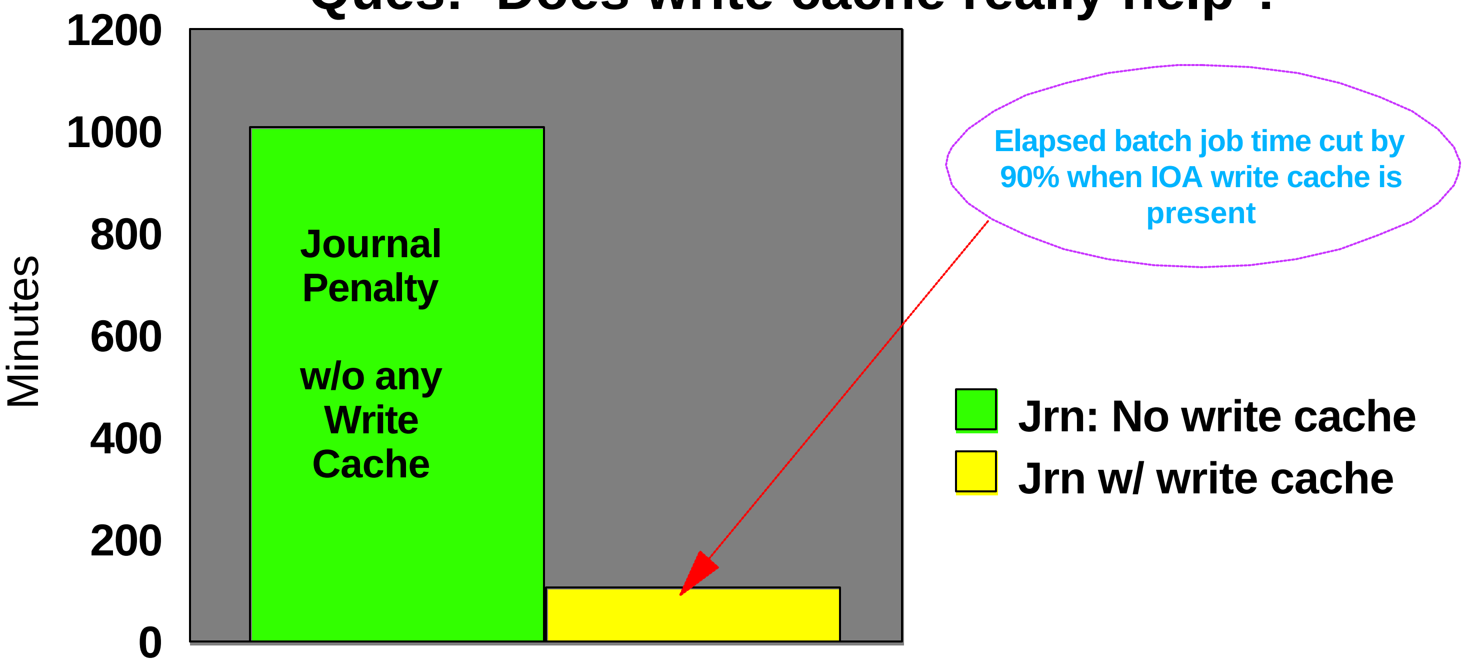
Write cache comes in a variety of sizes. How much difference does write cache make to Jrn Perf ?

IOA Write cache performance benefit

Elapsed Time

Batch Job

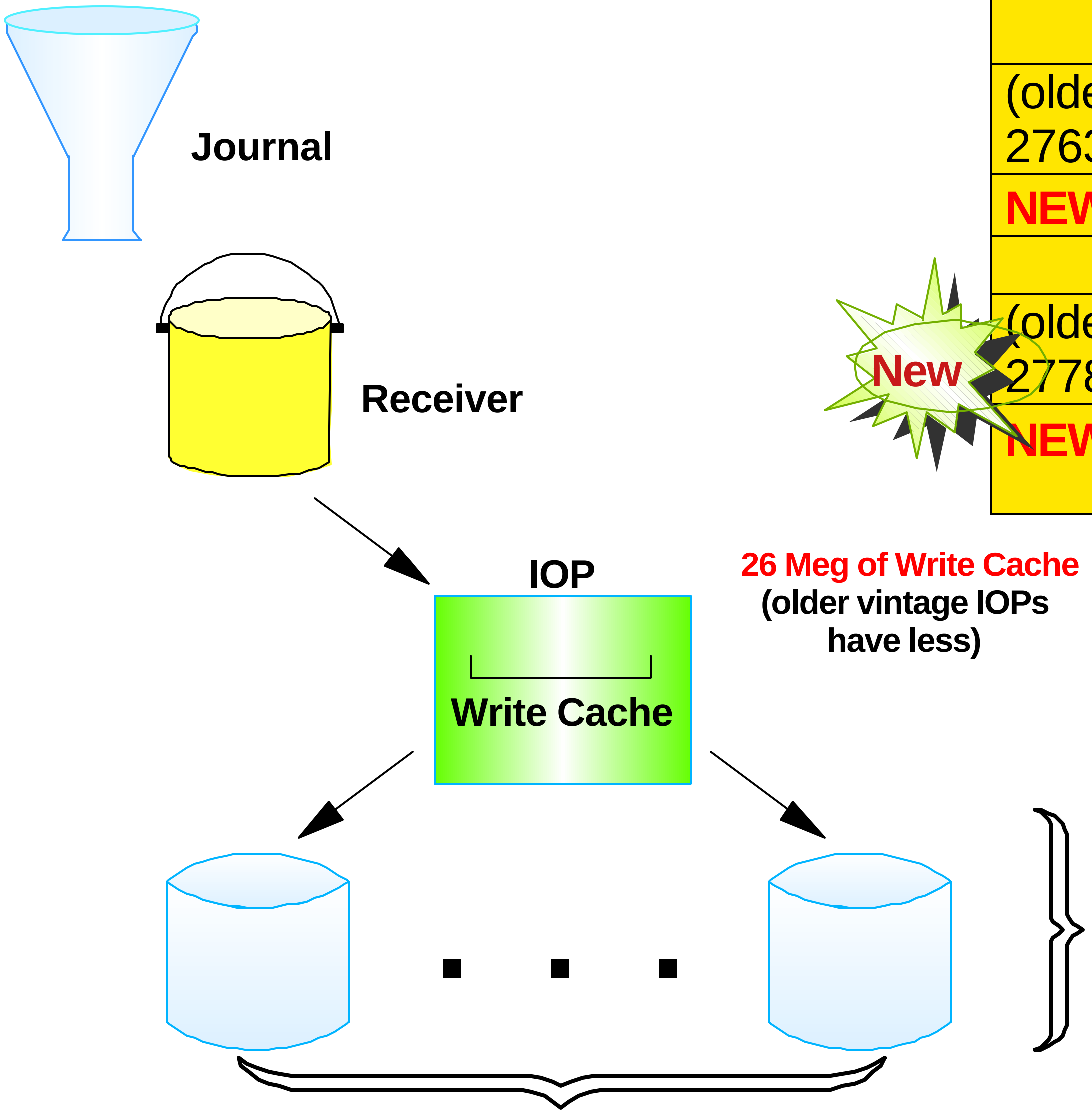
Ques: Does write cache really help ?



Elapsed batch job time cut by 90% when IOA write cache is present

Conclusion:

Supplying sufficient IOA write cache is probably the **most influential hardware decision** you can make in order to reduce journal overhead. It's more important than RAID vs. Mirroring, more important than User ASP vs. Sys ASP, more important than number of disk arms, more important than 7.2k RPM vs. 10k RPM disk drive speed.



Journal

Receiver

IOP

Write Cache

(Mirrored or RAIDed ?)

Need enough Arms

26 Meg of Write Cache
(older vintage IOPs
have less)

New

Model	Write Cache
(older) 2763	10 Meg
NEW !! 2782	40 Meg
(older) 2778	26 (104) Meg
NEW !! 2757	235 (757) Meg

Feb '03

Rule of thumb:

Get your hands on the maximum amount of IOA write cache you can afford.

26 Meg is often adequate, 4 Meg generally is not.

At rates approaching 1 Million Jrn Entries/Minute:
You need even more ! (The 757 Meg IOA helps !)

If you perform <u>lots</u> of journaling, avoid configuring the maximum (15) allowable drives per IOA
(because journal can swamp the write cache).

Limit yourself to 4 disks per IOA if you've got only 4 Meg
and 10 disks per IOA if you've got 26 Meg.

Summary of Configuration Choices

- **Spread** User ASP disks over multiple IOPs
  - The extra write cache helps
- Install faster disk arms (10K RPM)
  - Tends to reduce elapsed batch time by upwards of 30%
- Install larger IOP write cache (Model 2748 or newer)
- Add sufficient disk arms to the Journal ASP (up to 15)
  - ▸ Don't just focus on disk capacity, think about bandwidth and parallelism
- Specify ***MaxOpt1** to get the broadest arm usage

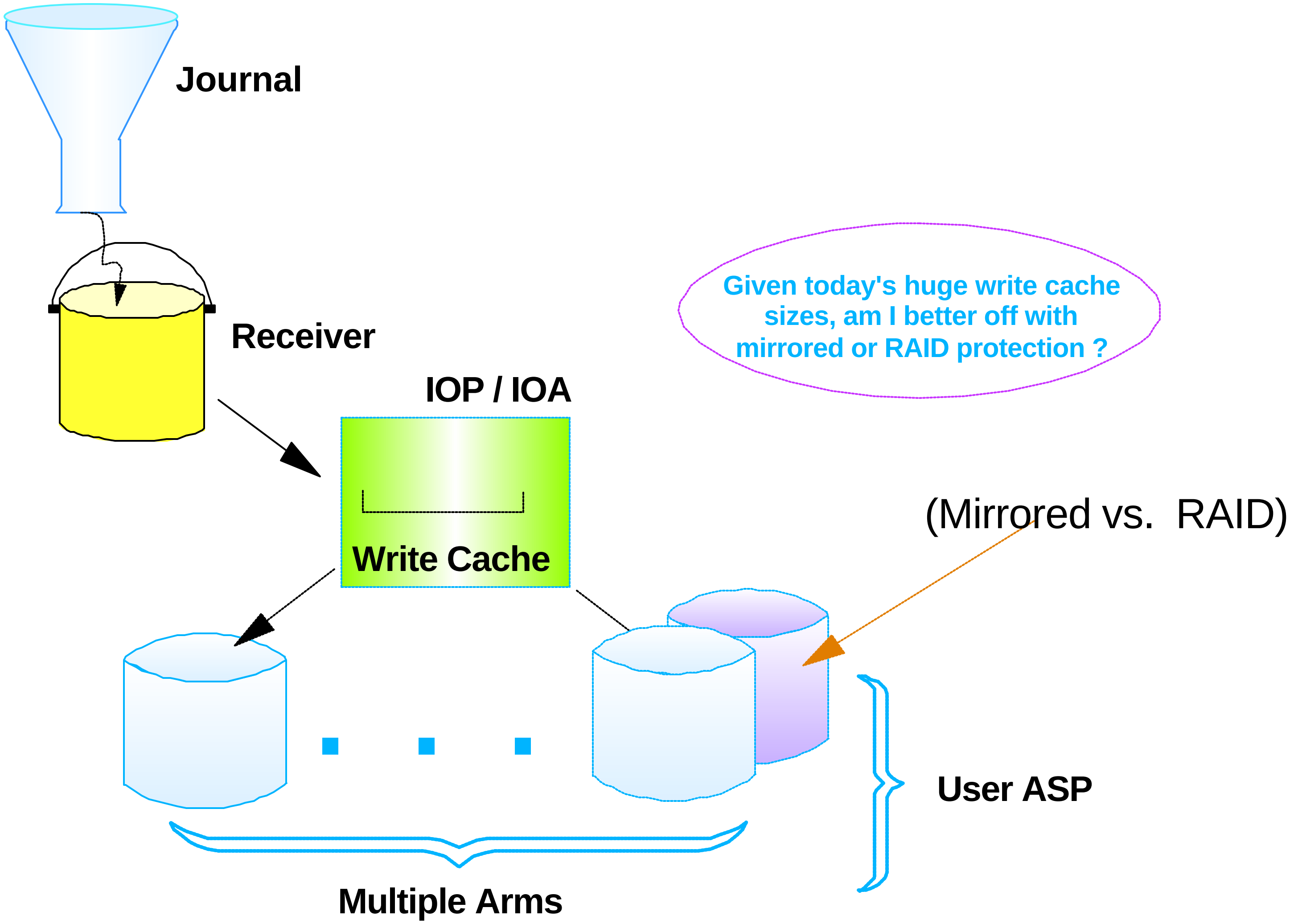
Notes:

Faster spinning disk drives definitely make a noticeable difference in disk-write intensive environments. Journaling activity (especially under a batch job) can clearly be such an environment.

If you have multiple vintages of disk drives in your shop, give the journal the best of the best -- he needs them.

Mirroring vs. RAID
Which is most Journal friendly ?

Are Mirrored drives still the best performance choice ?



Journal

Receiver

IOP / IOA

Write Cache

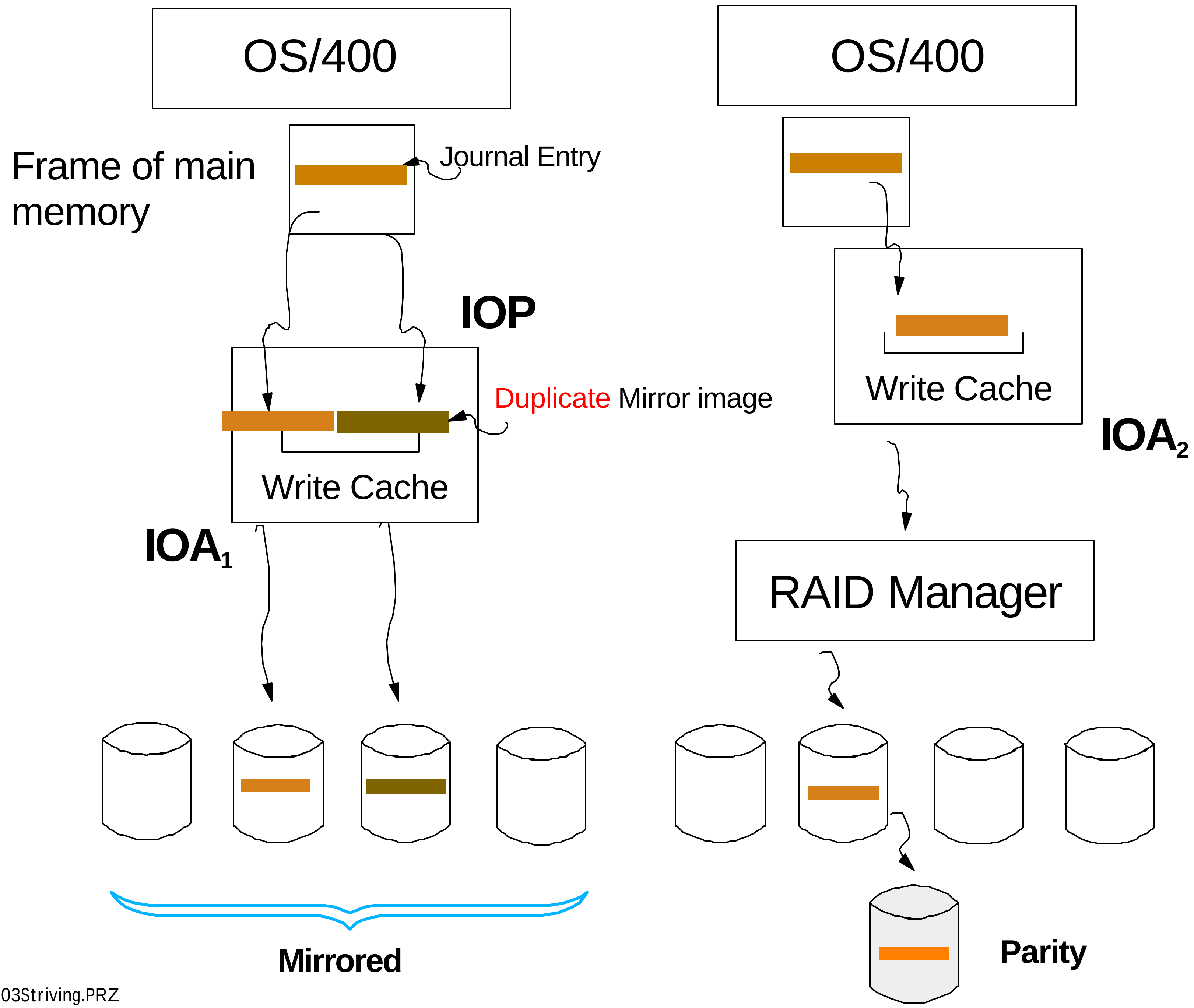
Given today's huge write cache sizes, am I better off with mirrored or RAID protection ?

(Mirrored vs. RAID)

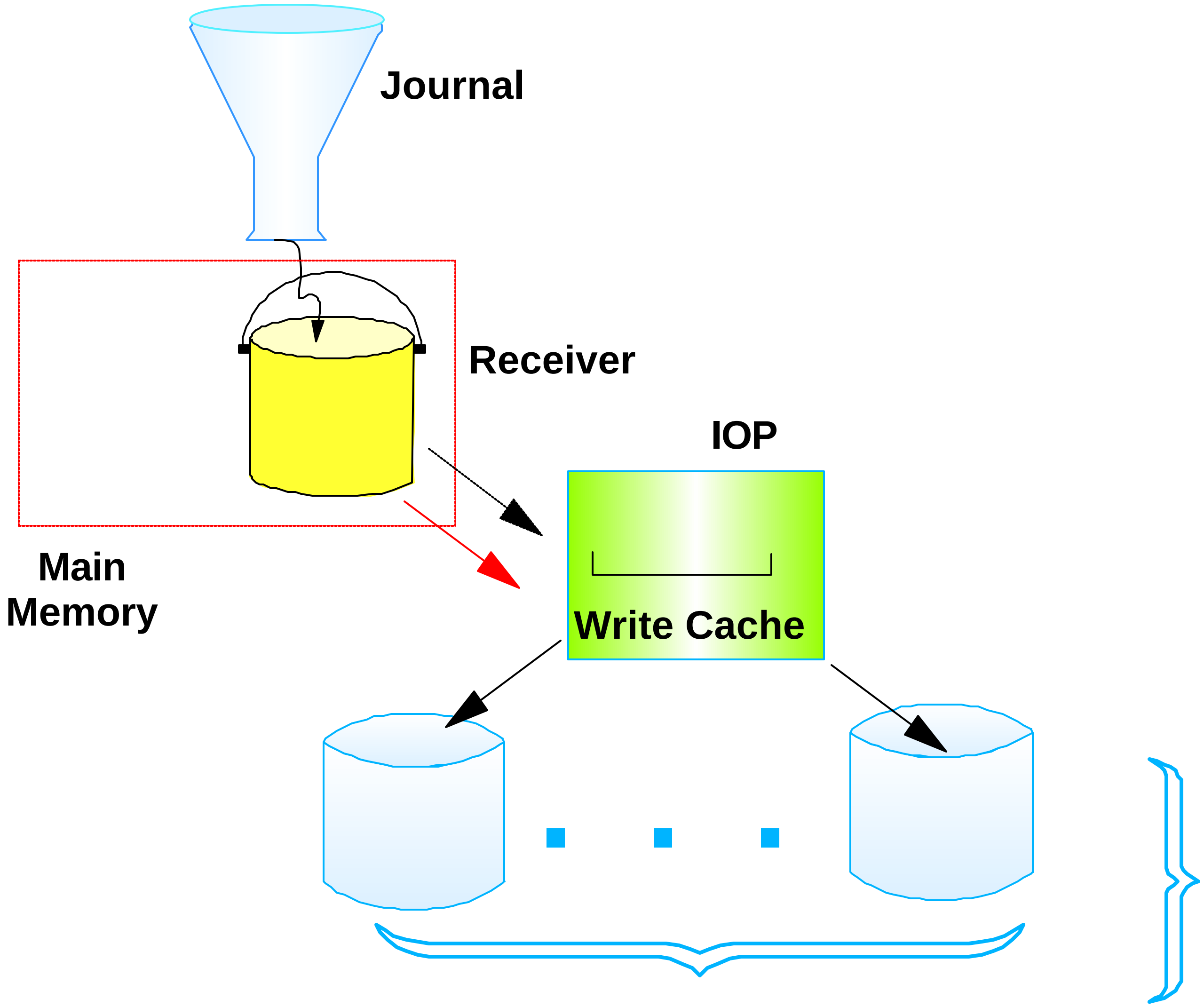
User ASP

Multiple Arms

Writes Required for Mirror vs. RAID



Use of write cache -- Extra Mirrored writes



User ASP

(Provided you have enough Write Cache, RAID5 drives may actually perform better for journaling than Mirrored drives)

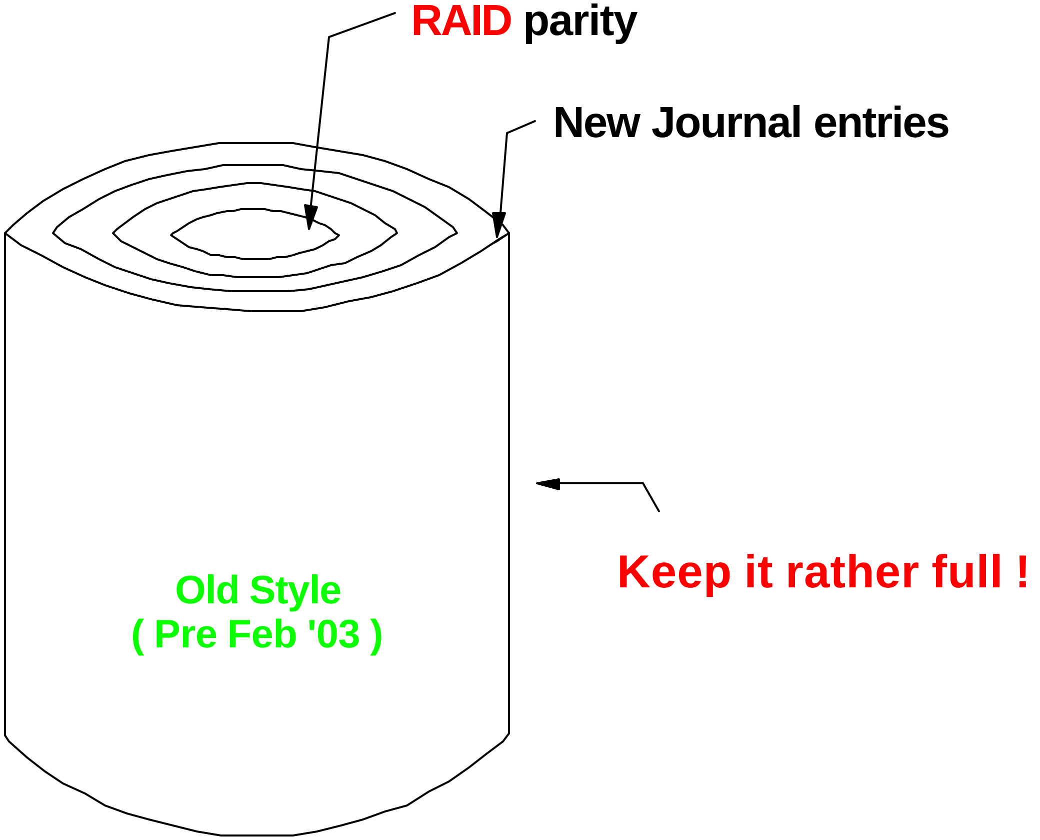
Notes:

Although former in-house tests on machines with limited quantities of write cache had demonstrated that mirrored disk drives can absorb heavy write-intensive journal traffic faster than RAID5 protected disk drives, the presence of the newest IOPs with 26 Meg (or more) of write cache has reversed that distinction.

Yes, I'd still prefer mirrored disk drives if you can afford them, but if that's not practical at least strive for use of IOPs with large quantities of write cache associated with the disk drives housing your journal receivers.

From a journal perspective the speed of your disk drives (10k RPM) and the size of your write cache may be the most important hardware choices you can make if you're striving for optimal performance.

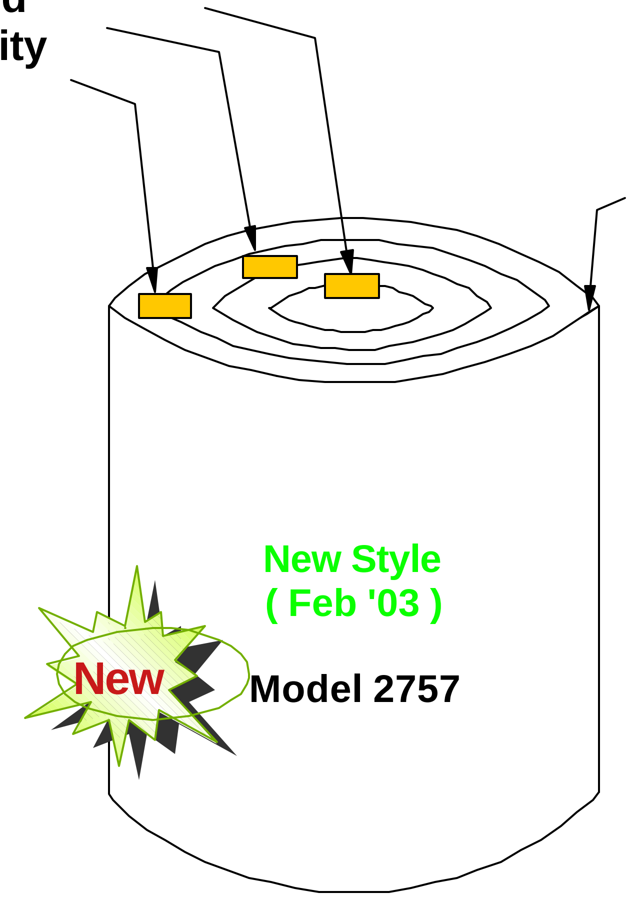
The dreaded arm sweep



Old-style RAID imposes up to a 2.2 fold performance penalty
on Journal if you don't have enough write cache.

Better performance

Disbursed RAID parity



New Journal entries

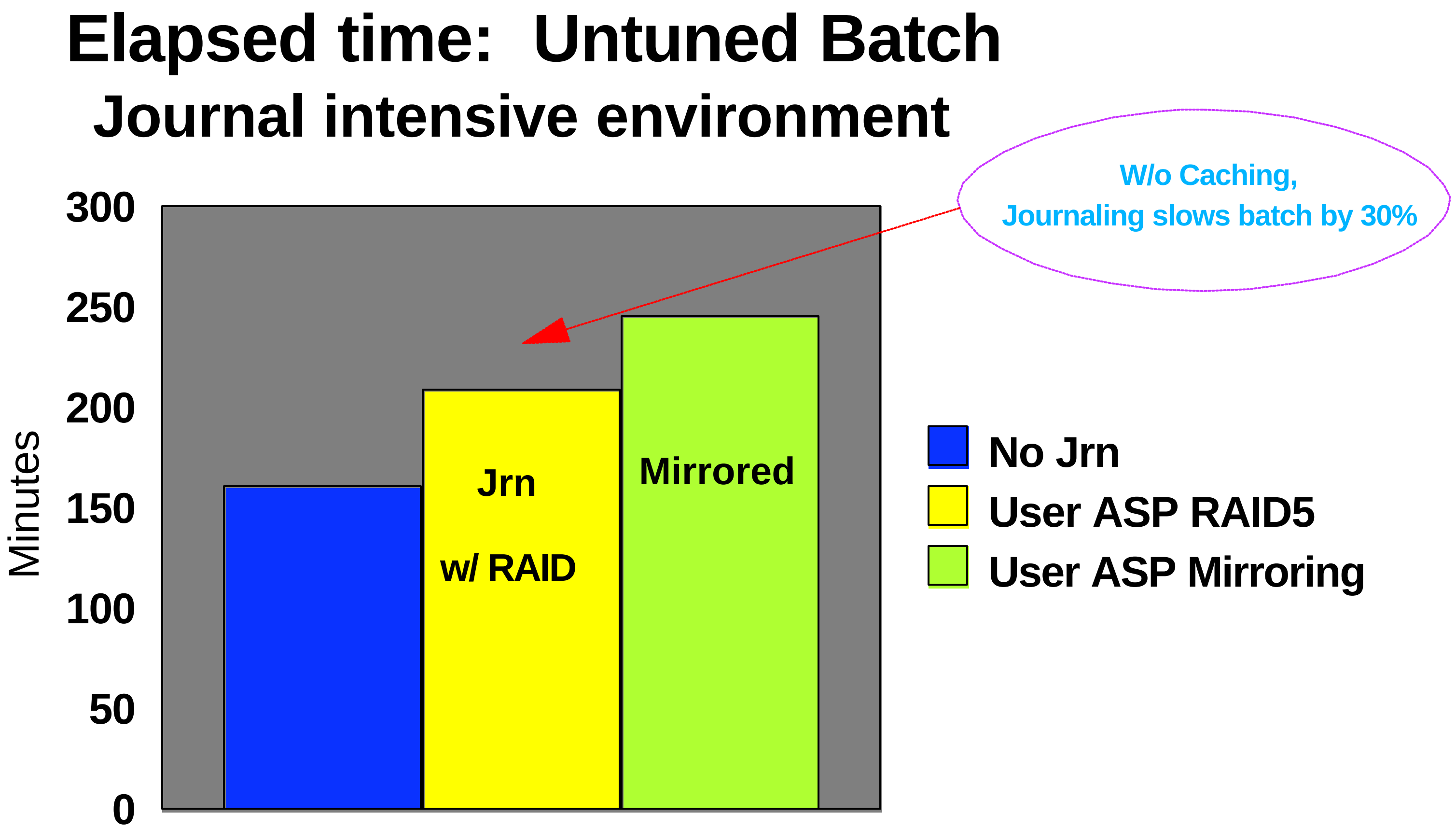
New Style
(Feb '03)

New

Model 2757

V5R2!

Mirroring impact on elapsed time to Journal



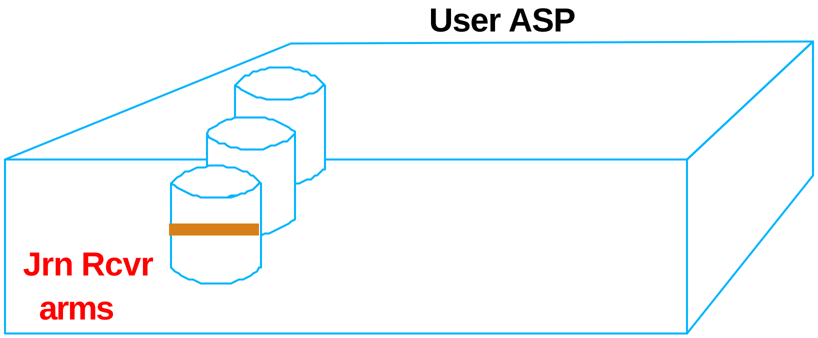
Conclusion: Extra overhead and CPU pathlength to write second copy of each main memory page frame on behalf of mirrored User ASP vs. RAID-5 User ASP slowed this batch job by **53%**.

RAID-5 puts less pressure on the write cache (slows batch by only **30%**).

Clearly this is an environment in which use of Jrn Caching ought to be used to further reduce journal overhead.

What About Auxiliary Storage Pools?

- Likely to improve performance when journaling
  - Provided you configure enough disk arms (1 is rarely enough !)
  - Place your journal receiver in a **user ASP**
    - CRTJRNRCV JRNRCV(RECV0001) ASP(2)
- Try to avoid placing multiple "active" receivers in the same user ASP
    - (Unless you've got LOTS of disk arms in this ASP)



Notes:

Auxiliary Storage Pools (ASPs) represent an excellent opportunity to give the journal receiver a set of **private** disk arms for his sole use.

Given enough disk arms in such a pool, your journal receiver will spread itself across the arms and achieve efficiency by writing to multiple disk arms in **parallel**. The more arms the higher the bandwidth available to the journal receiver and hence the higher the disk traffic he can service without slowing you down.

If you give him too few disks arms, however (and one arm is rarely enough) the use of User ASPs can be counterproductive. I prefer to see at least three disk arms in each User ASP.

How Many Disk Arms Are Enough?

- One arm per User ASP is rarely enough for best performance
- Formerly
  - Used up to 10 primary (fastest) arms per receiver
  - Plus 5 additional arms if **RcvSizOpt** (***RmvIntEnt**)
- A **round robin** algorithm is employed
- You can increase this maximum to 100 arms
  - Provided you specify **CHGJRN** ... RcvSizOpt(***MaxOpt1**)

Journal tuning parameters

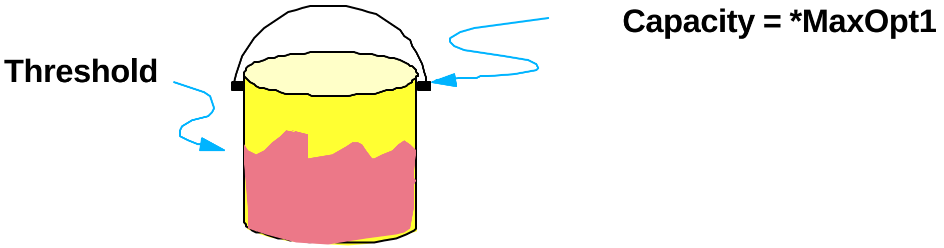
Journal Receiver thresholds

- **How often should I change receivers ?**

  - Consider use of a threshold

  - **CRTJRNRCV** ... **THRESHOLD**(1800000)   { Means 1.8 Gig }
    - Note: Threshold is in units of Kilobytes
    - Higher your threshold:
      - More disk arms we use
      - More bandwidth you achieve

  - If your applications produce a Gig or more of journal traffic per hour you'll probably want to step up to a ***MAXOPT1** setting for your journal.



Notes:

You get to select both the maximum journal receiver size and a corresponding threshold value which alerts you when the receiver is approaching its maximum size. Make these choices wisely!

You can elect to move away from the traditional 2 Gig maximum journal receiver size and can step up, instead, to the new 1 TB receiver size. Be sure to remember to similarly increase your threshold to a comparable value if you want to shoot for some total journal receiver capacity greater than 2 Gig.

By increasing both the threshold and capacity you're decreasing the frequency with which journal receivers need to be swapped -- and hence reducing the instances of any performance spike associated with the Flush which accompanies such a swap.

Increased Journal Capacity

- Larger receiver capacity
  - Default 2 Gig receiver capacity can be increased to 1 TB
  - Requires *maxopt1 option on CHGJRN or CRTJRN
  - 2 Billion Sequence Numbers climbs to nearly 10 billion

- May want to select larger journal receiver threshold when you create the receiver
  - To employ a threshold larger than 2 Gig you must also request a larger max jrn size

- Example:
  - CRTJRNRCV JrnRcv(MyLib/Rcv2) Threshold(70000000) { 70 Gig }
  - CRTJRN Jrn(MyLib/Jrn2) JrnRcv(MyLib/Rcv2) RcvSizOpt(*MaxOpt1) { Rqsts 1 TB max }

Notes:

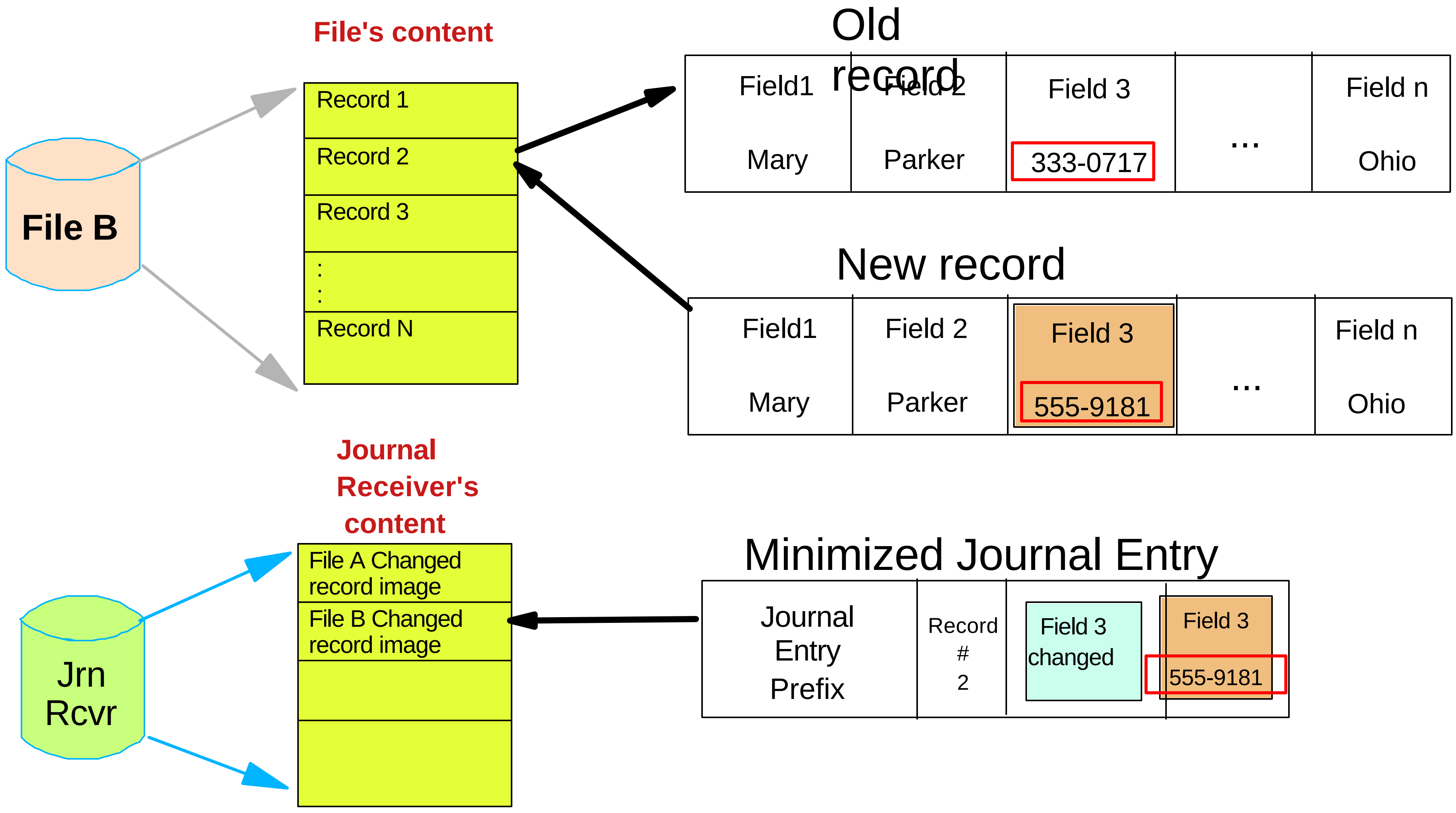
The larger capacity (1TB) journal receiver size is designated by specifying RcvSizOpt(***MaxOpt1**) on your CHGJRN command.

You can step up to ***MaxOpt2** (which not only gives you the full 1 TB size limit but also allows individual journal entries to be wider when necessary -- a need you may face if you elect to store large BLOBs in your database files or SQL tables).

Besides the 1 TB capacity, both *MaxOpt1 and *MaxOpt2 bump the maximum number of journal sequence numbers allowed by nearly 5-fold.

Minimizing Qty of Jrn Data

Minimized-Data Journal entry content



File B

File's content

- Record 1
- Record 2
- Record 3
- :
- :
- Record N

Old record

Field1	Field 2	Field 3	...	Field n
Mary	Parker	333-0717		Ohio

New record

Field1	Field 2	Field 3	...	Field n
Mary	Parker	555-9181		Ohio

Jrn Rcvr

Journal Receiver's content

- File A Changed record image
- File B Changed record image

Minimized Journal Entry

Journal Entry Prefix	Record # 2	Field 3 changed	Field 3 555-9181

Journaling Minimal Data

Journal **Minimal Data** is an option affecting how much space updated Database records/rows consume in your journal receivers.

With Journal Minimal Data enabled only the essential changed bytes are written to the journal (not the whole record image).

This can lead to:

- **Reduction in the size of journal receivers**
- Reduction in the frequency for journal housekeeping (i.e. swapping receivers).
- Reduced comm line traffic in a remote replication environment utilizing Rmt Jrn support since minimized journal entries consume less bandwidth enroute to the remote machine

Customer experience with Journal Minimal Data

*CHGJRN ... MINENTDTA(*FILE)*

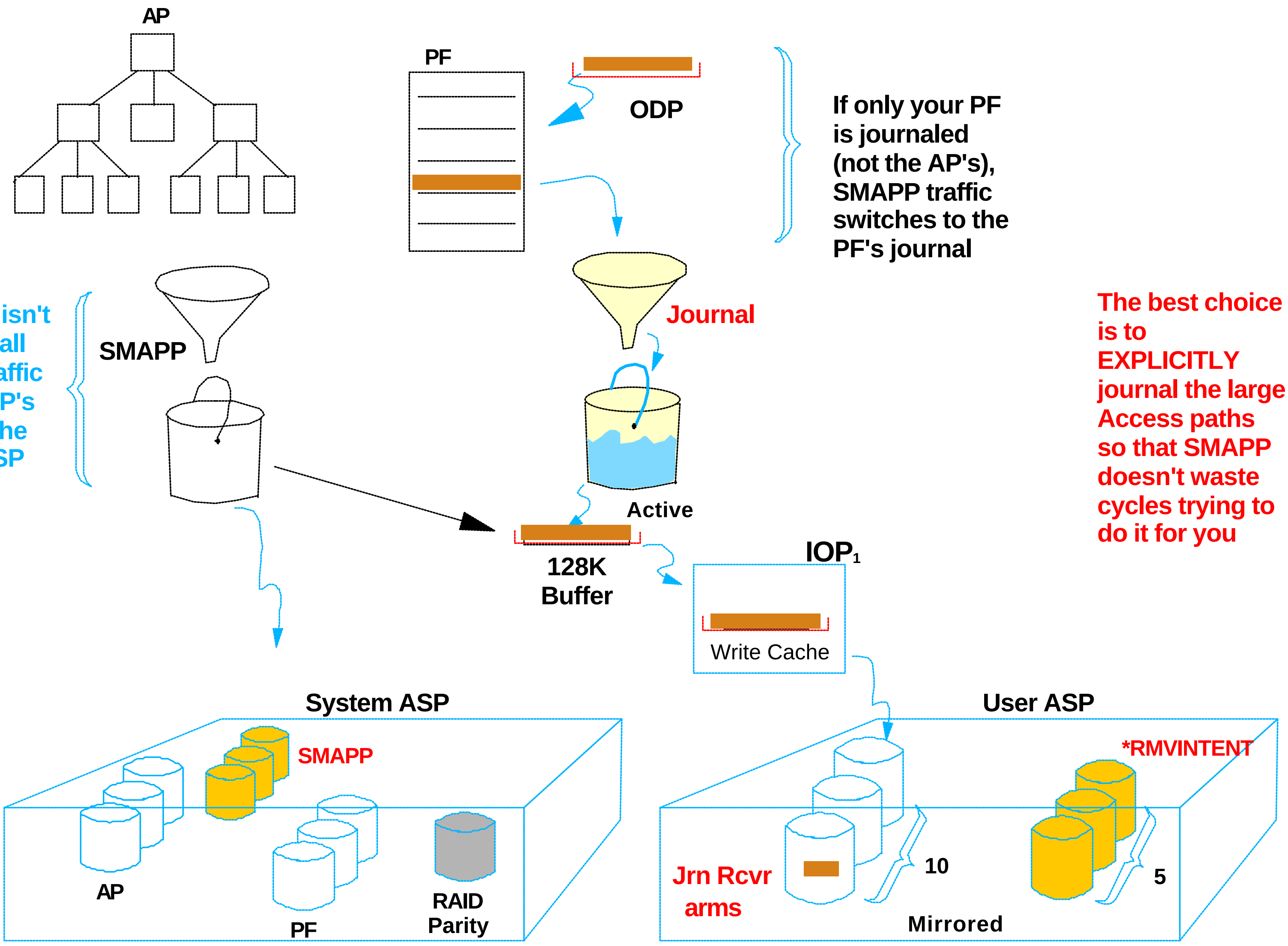
ERP Environment

	Ordinary	Jrn_Min	Difference
Size Jrn Ent	637 bytes	411 bytes	35% less
Disk writes	55/sec	36/sec	34% less
Write width	16k/write	12k	25% less
CPU busy	61.5%	60.6%	1.5% less

Beat SMAPP to the punch

Explicitly journal your large Access paths (STRJRNAP)

If your PF isn't journaled all SMAPP traffic for your AP's heads to the system ASP



If only your PF is journaled (not the AP's), SMAPP traffic switches to the PF's journal

The best choice is to EXPLICITLY journal the large Access paths so that SMAPP doesn't waste cycles trying to do it for you

Throttling the SMAPP Overhead

- How does SMAPP affect performance?
  - SLIC provides background SMAPP jobs
    - **Some evaluate the eligibility of Access Paths for protection**
    - **Some re-tune the protection**
    - **Some start and stop implicit journaling for the selected access paths**
    - **Some sweep changed pages from memory**
    - **All consume CPU cycles**
  - SMAPP becomes even more aggressive
    - 150 -> 120 -> 90 -> 70
    - **You may want to tone it down on your target/back-up system (EDTRCYAP)**
- What can you do about it?
  - **Explicitly journal large, frequently modified access paths**
    - Moves disk traffic from system ASP disk arms to user ASP disk arms
    - **Cuts down on the costs of reevaluation decisions in the background**
  - **CHGJRN** RCVSIZOPT (*RMVINTENT)
    - **Overlaps access path journal disk writes with PF journal disk writes**
    - **Reduces amount of communication line traffic for Remote Journal**

Notes:

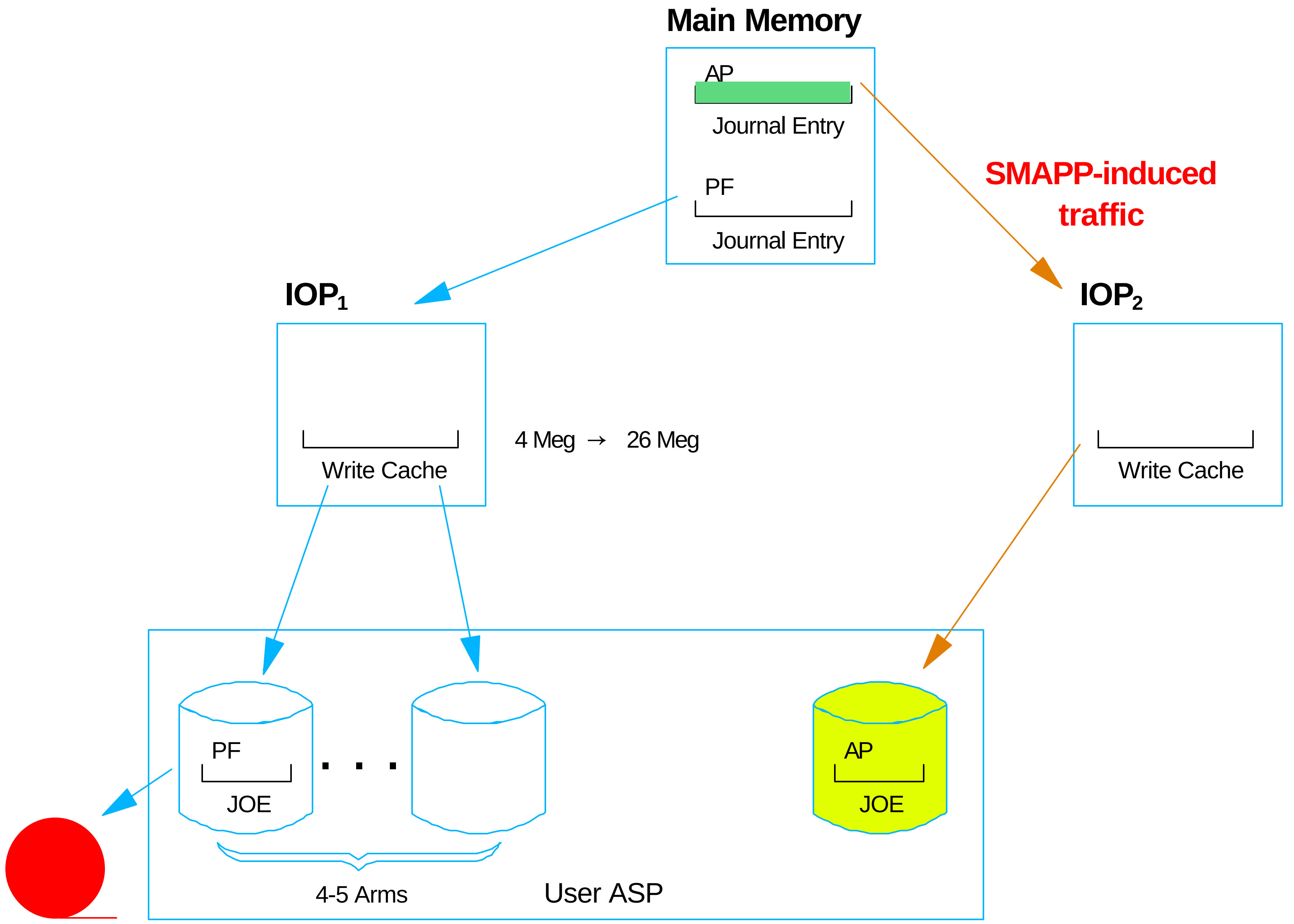
SMAPP (System Managed Access Path protection) is always running in the background trying to identify and protect (i.e. implicitly journal) your largest access paths so that your abnormal IPL time doesn't go through the roof. That's a darn good thing.

But... it takes some CPU cycles to keep making this decision on your behalf. If you know with certainty which particular access paths are both huge and important to your business, you can reduce some of this background performance penalty by making the decision for us instead of forcing the operating system to make the decision day after day after day each time you open or close a file.

Net: Employ STRJRNAP to identify the specific access paths which you know you'll want access to quickly if the machine were to crash. By doing so, we'll burn fewer cycles trying to make that decision in the background on your behalf.

Putting a fork in the road

Overlapped Writes With *RMVINTENT



Notes:

The RcvSizOpt(*RmvIntEnt) option on the CHGJRN command is a winner. Use it.

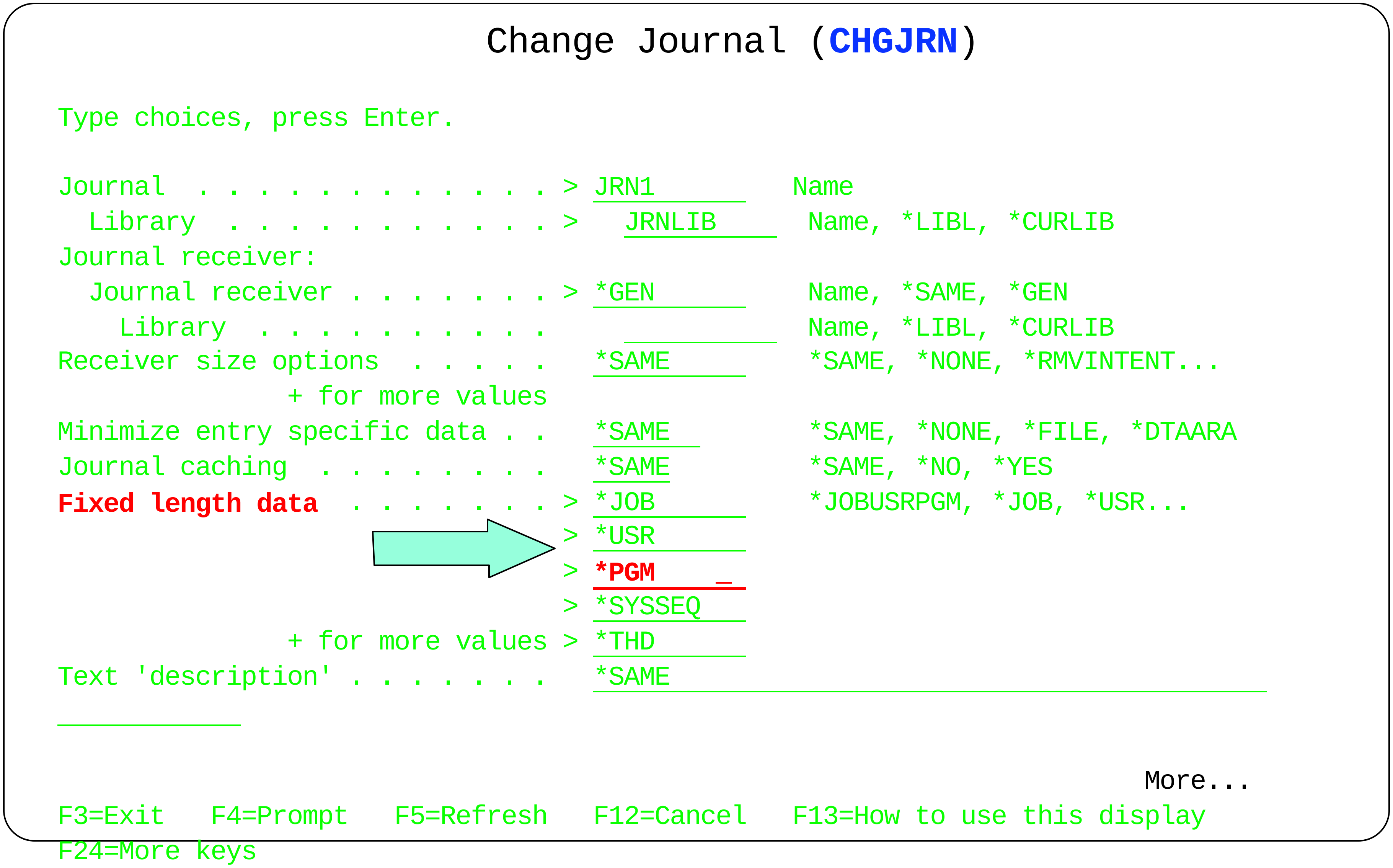
It gives the operating system permission to put a fork in the road and write your application-generated journal entries on behalf of database changes to one set of disk arms while writing the system-generated hidden journal entries (useful only to IPL processing) to a separate set of disk drives.

When SMAPP is enabled, this means your PF journal images go to one set of disk arms, and your AP journal images go to another -- but don't worry, IPL knows how to find and use both.

This physical separation of journal entries is generally a wise choice. The one instance in which you'd be better off letting them commingle to the same set of disk arms is when you have only one disk arm configured in your User ASP.   If you've got only one disk arm, ignore this option. If you've got three or more disk arms in the User ASP and lots of large access paths, it's probably a winner.

Customizing the Chaff

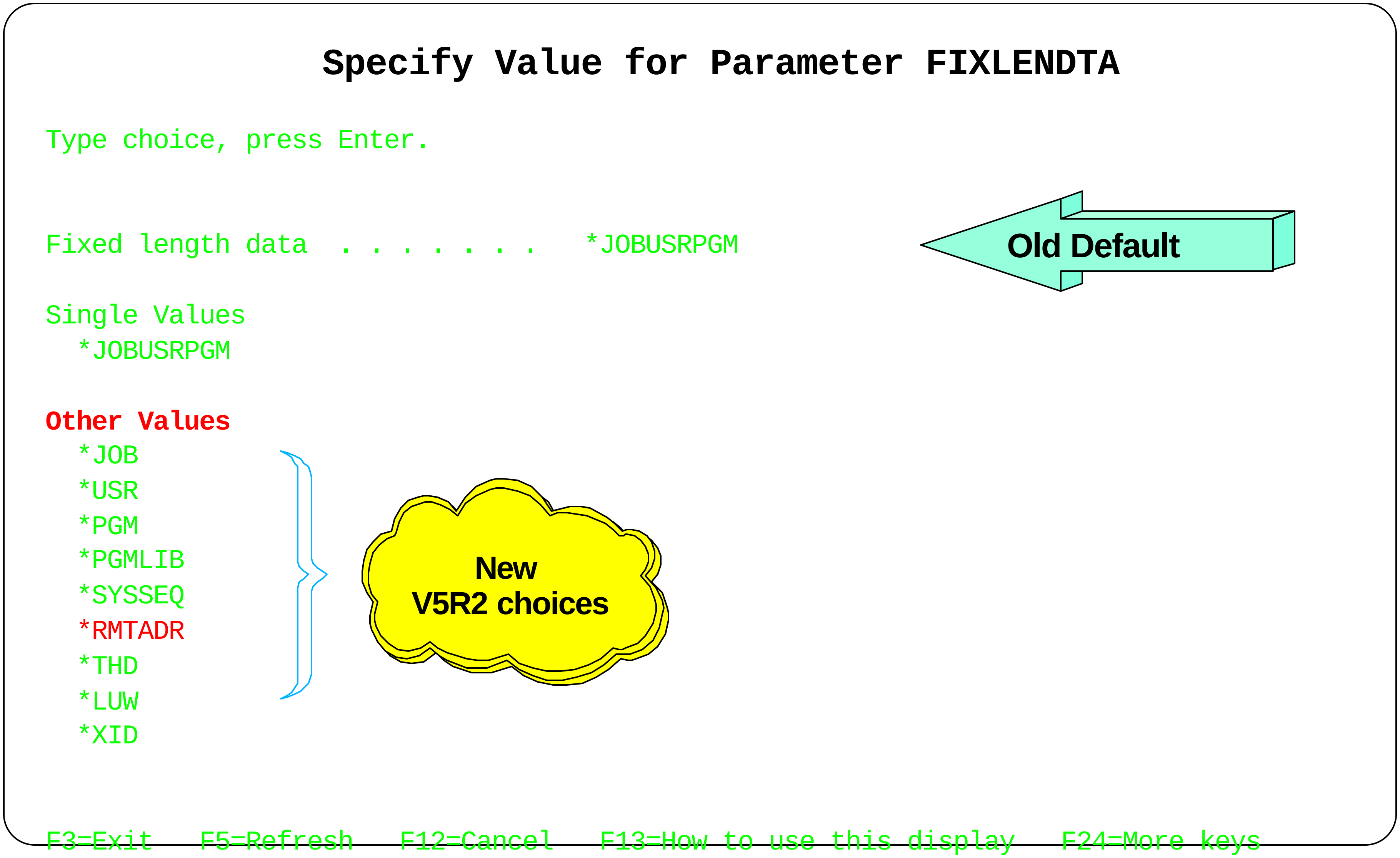
Customizing what we collect



```
                        Change Journal (CHGJRN)

Type choices, press Enter.

Journal . . . . . . . . . . . . . > JRN1          Name
  Library . . . . . . . . . . . . >   JRNLIB        Name, *LIBL, *CURLIB
Journal receiver:
  Journal receiver . . . . . . . > *GEN            Name, *SAME, *GEN
    Library . . . . . . . . . .                     Name, *LIBL, *CURLIB
Receiver size options . . . . .   *SAME             *SAME, *NONE, *RMVINTENT...
               + for more values
Minimize entry specific data . .  *SAME             *SAME, *NONE, *FILE, *DTAARA
Journal caching . . . . . . . .   *SAME             *SAME, *NO, *YES
Fixed length data . . . . . . . > *JOB              *JOBUSRPGM, *JOB, *USR...
                                > *USR
                                > *PGM    _
                                > *SYSSEQ
               + for more values > *THD
Text 'description' . . . . . . .  *SAME
____________

                                                                       More...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

CHGJRN command with new FIXLENDTA parm

CHGJRN . . . FIXLENDTA(*RMTADR)



```
                 Specify Value for Parameter FIXLENDTA

Type choice, press Enter.


Fixed length data  . . . . . . .   *JOBUSRPGM

Single Values
  *JOBUSRPGM

Other Values
  *JOB
  *USR
  *PGM
  *PGMLIB
  *SYSSEQ
  *RMTADR
  *THD
  *LUW
  *XID


F3=Exit   F5=Refresh   F12=Cancel   F13=How to use this display   F24=More keys
```

All options displayed for FIXLENDTA Parameter

Be Selective

★ Advise us to collect only what you truly need,
no more and no less

Summary

Which Journaling Parameters Can Help?

- **CHGJRN . . .**
  - Use of **MINENTDTA**
    - **Reduces quantity of bytes written to the journal receiver (only changed bytes written)**
    - **Reduced size of Jrn Receiver 35% in one shop**
  - RcvSizOpt (***RmvIntEn**t)
    - **Routes hidden/implicit internal journal entries to a separate set of disk arms**
    - Recycles disk space
  - RcvSizOpt (***MinFixLen**)
    - **Reduces quantity of bytes written and CPU consumed per journal entry**
    - **Saved 5% time and space in one shop**

Notes:

These are tuning choices you face regarding your journal itself.

Employing *MinFixLen is a useful way to reduce both CPU consumption and disk space by weeding out the pure auditing information which otherwise accompanies each and every journal entry. If your auditors don't need it, why collect it ?

MINENTDTA can be an excellent choice if you have applications which update only a few fields in each record. (Why make the journal capture the WHOLE record image if only a few fields change ?)

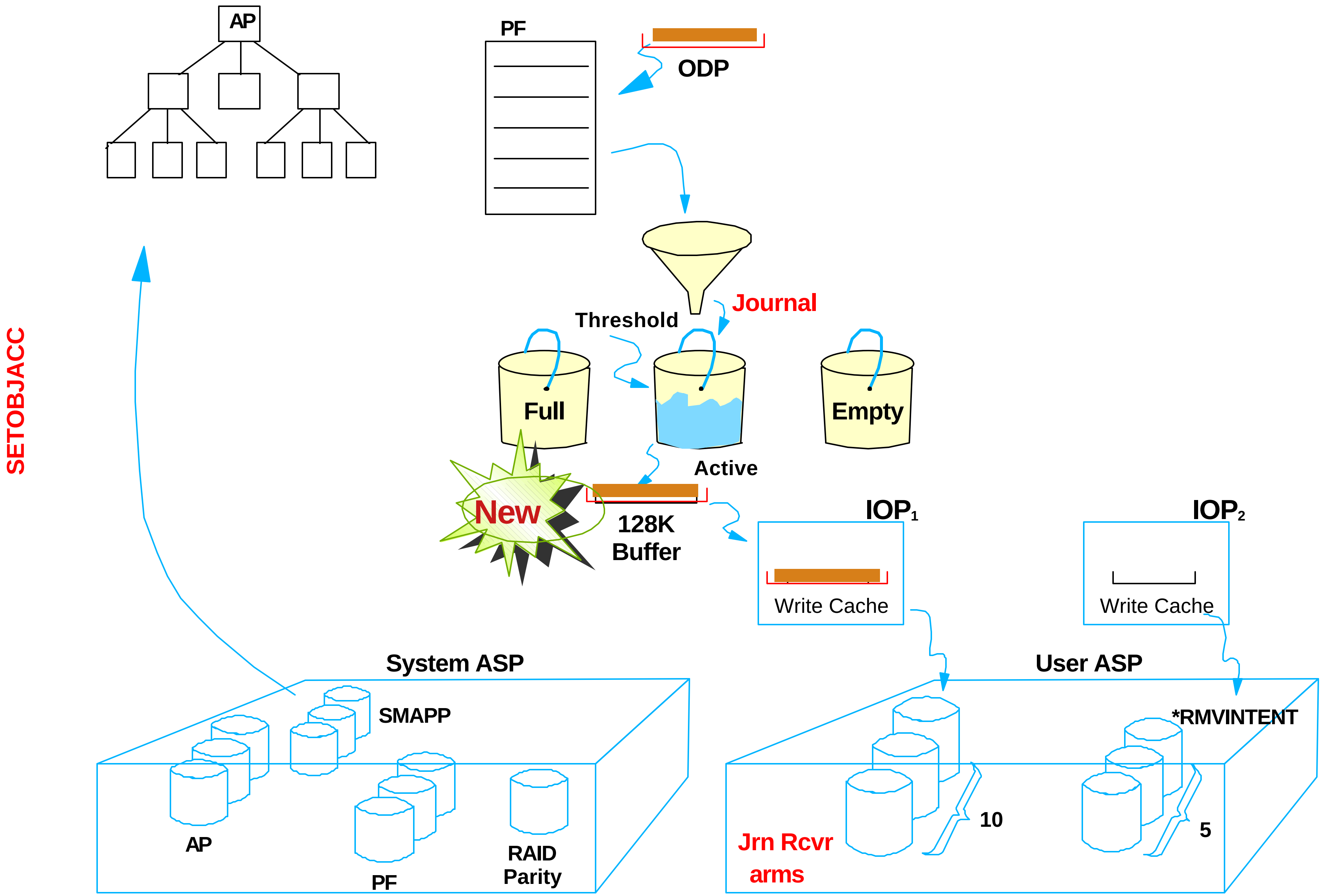
Aggressive Journal Cache PRPQ (OS/400 option in V5R2)

V5R2!

Notes:

This is a new journal performance offering.
It's especially effective for speeding up batch jobs.

The Journal environment - - Caching at three levels



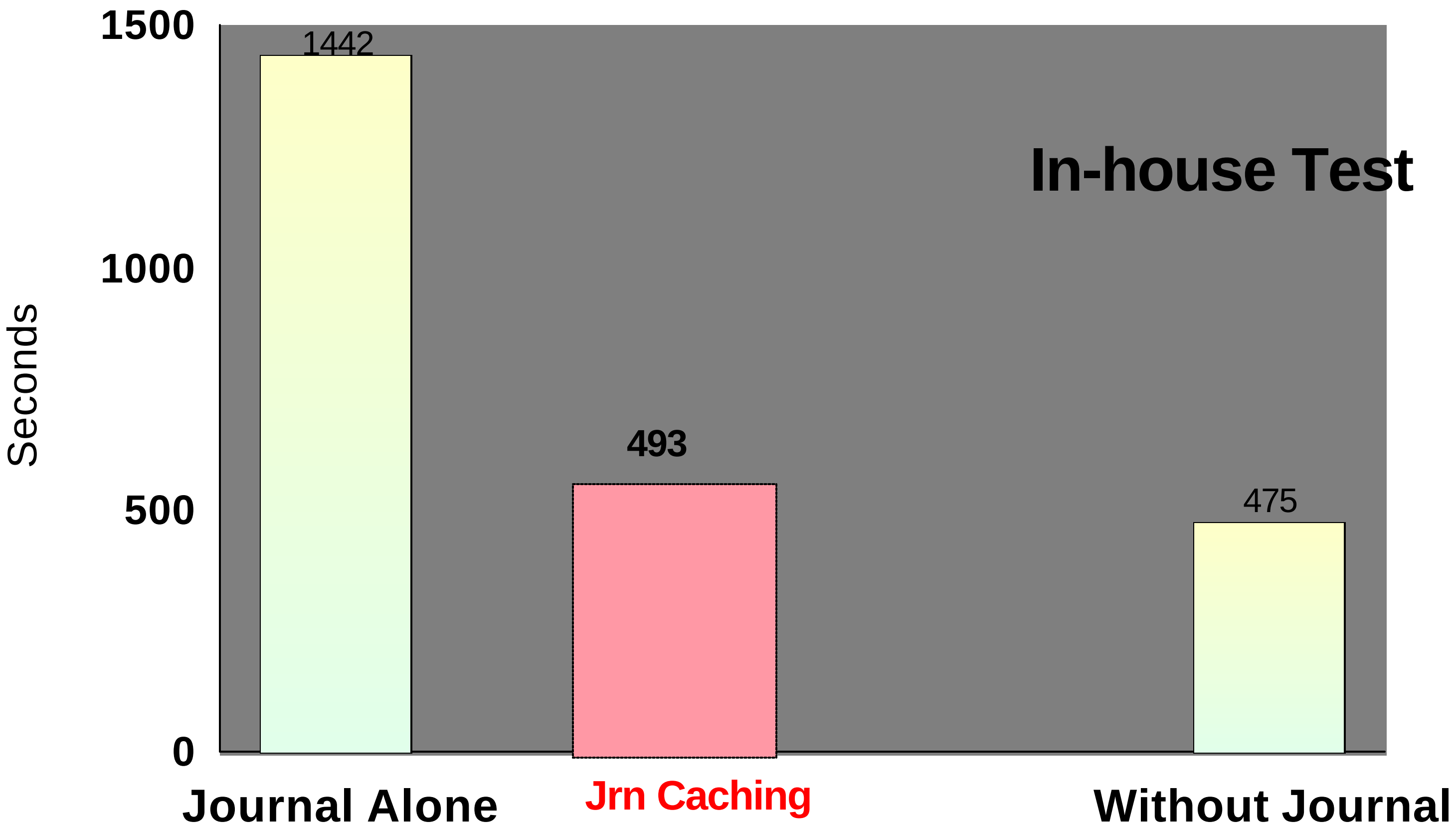
Notes:

This is the "big picture". It summarizes much of what we'll talk about, from your opportunity and responsibility to tune the use of the ODP buffer all the way down to the selection of the proper write cache size in the IOPs.

Spreading your disk arms within a User ASP across sufficient IOPs so as to maximize use of write cache is an important configuration choice you can make. My rule-of-thumb is that disk write intensive operations like journaling perform best when old-style IOPs (those housing 4 Meg of write cache or less) service no more than 5 disk arms. The newer vintage IOPs (those with 26 Meg or more of write cache) can easily handle up to 10 disk arms apiece.
Above 15 arms gets dicey since high volume journal environments may eventually get starved for sufficient write cache.

Net: Your hardware configuration does matter!

Impact of the Jrn Caching



DB Updates during Batch

Notes:

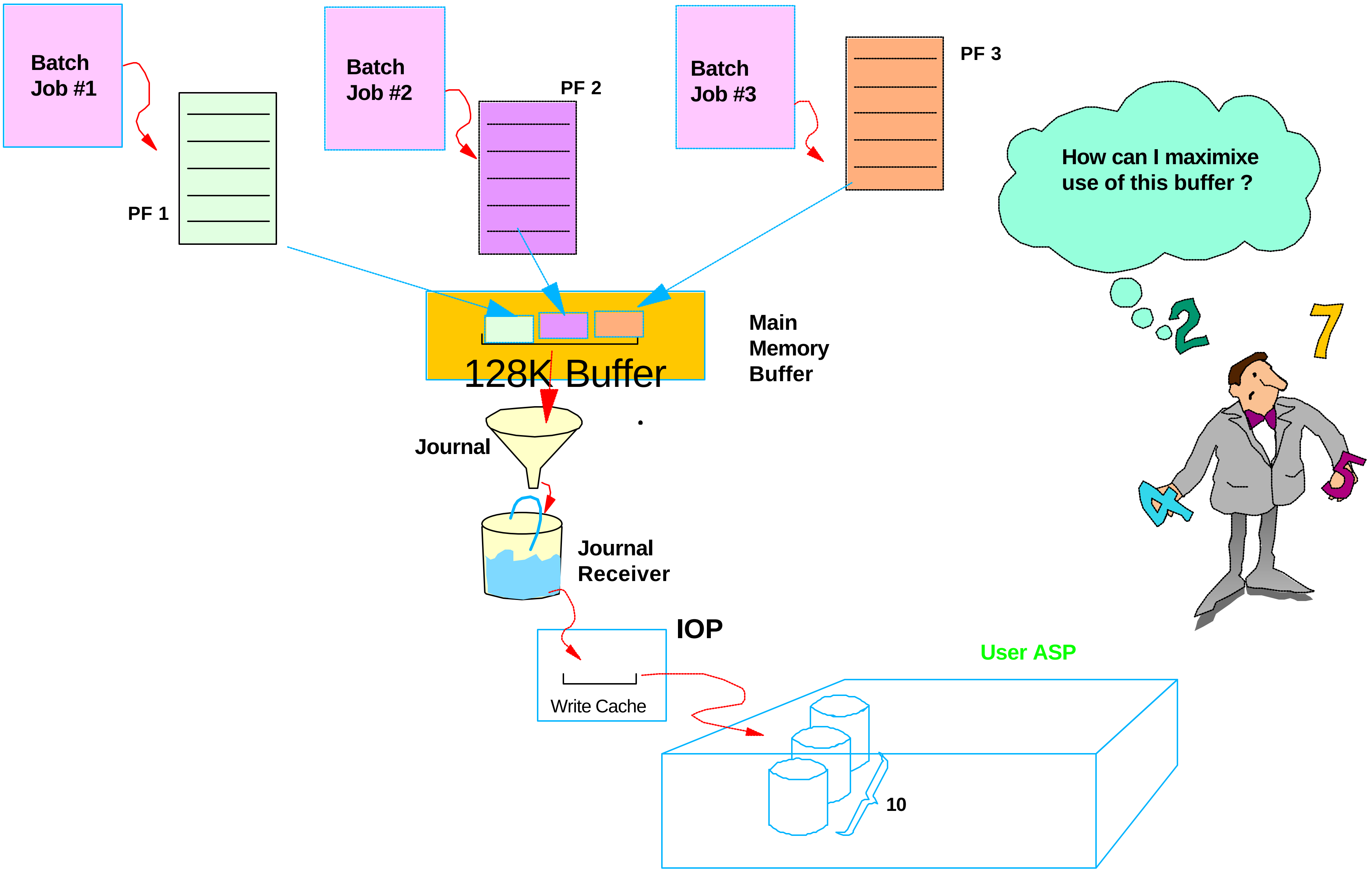
While there are lots of software things you can tinker with in an effort to reduce journal performance overhead, this one is head and shoulders above the rest.

If you've got the Journal Caching option installed, use it. If you don't, get your hands on a copy.

It single-handedly can do in minutes what some folks struggle weeks or months to accomplish when attempting to coax good performance out of their journaling environment.

If you get nothing else from this talk, make yourself a mental note to search out and investigate this option.

Journal Bundling



Notes:

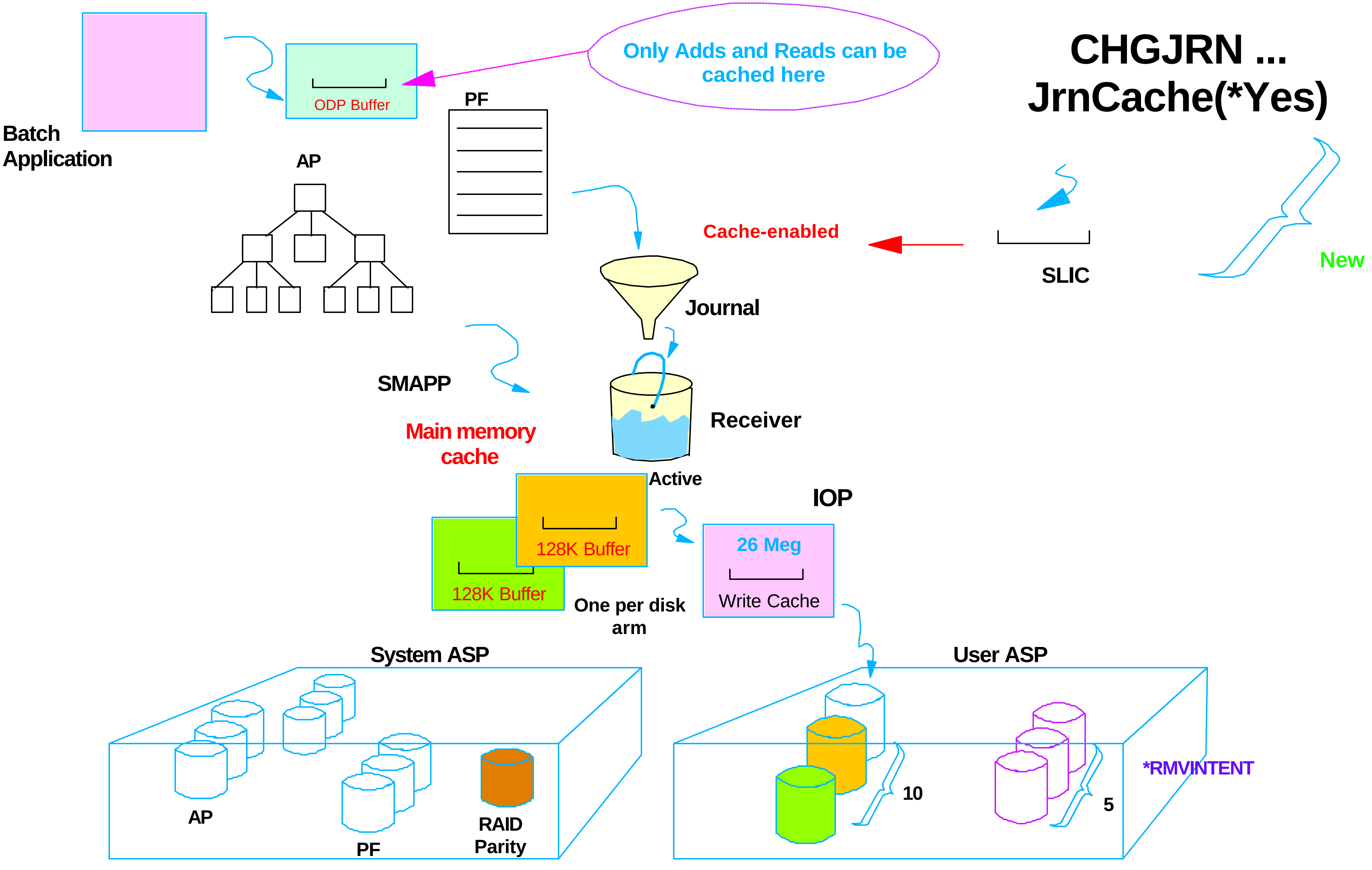
The SLIC layer of the journal microcode support gathers together journal entries provided by simultaneously operating jobs into a single main memory buffer.

The resulting set of journal entries are treated as a single string and are known as a bundle. This bundle is sent from main memory to the write cache of the IOA and ultimately to the disk surface in unison in a single disk rotation.

Such bundling behavior occurs naturally across jobs as the rate of journal entry creation speeds up. It can be further enhanced by installing and enabling the Journal Caching. With Journal Caching present, not only does such bundling behavior occur across jobs but also within a single batch job such that multiple separately updated SQL rows are bundled into a single journal string.

Such bundling brings about improved journal performance.

Journal Caching



Notes:

The new aggressive optimal caching behavior is initiated by enabling JrnCache(*Yes).

You need to enable this caching behavior individually for <u>each</u> **journal you want to speed up.**

The performance advantage stems from the fact that the caching option manages a set of main memory buffers, one for each disk arm across which your journal receiver is spread. Each buffer can be up to 128k in size, they come and go as needed, dynamically.

Journal Caching

- Primarily aimed at reducing journal overhead for long-running batch jobs which **don't** already employ commitment control
  - ➢ Increases main memory caching efficiency and effectiveness
  - ➢ Journal entries linger in main memory buffer until **optimal size** is achieved (128K)
  - ➢ Corresponding database records also linger in main memory
- Buffers only Database "Adds", "Updates", "Deletes"
- The first commit, or close operation flushes the buffer

Performance Benefits of Journal Caching

(Batch Job Without Benefit of Commitment Control

- 5 Million DB operations (10% Adds, 90% Updates)
- 9 Million resulting Journal entries

  (captured both before and after images)

	Elapsed Time	
Original Batch run, no Journaling	**1118 Sec**	**Base Run**
Ordinary Journaling enabled	**9773 Sec**	
With commitment control	**1593 Sec**	
Using the new Journal cache option	**1433 Sec**	

May also help **target** machine keep-up if files are journaled on the target side

Notes:

These are measured results for a batch job we ran in the lab. The batch job was provided by a customer who enlisted our assistance in speeding up their overnight batch runs.

For V5R2: The former Journal Caching PRPQ grows up !

- Impact becomes broader
  - ➢ It's not just for DB any more

- Becomes a priced feature of the Operating System
  - ➢ Appears on **CHGJRN** command

```
CHGJRN JRN(MYLIB/MYJRN) JRNCACHE(*YES)
```

V5R2!

Notes:

Notice that it's the high volume database operations such as ADD, UPDATE, and DELETE of records/rows which get buffered in the cache.

Both the database records themselves and the matching journal entries emitted on their behalf linger in main memory longer -- until the cache is full or until something ensues which requires us to flush the buffer to disk (closing the file initiates such a flush, so does issuing a commit verb).

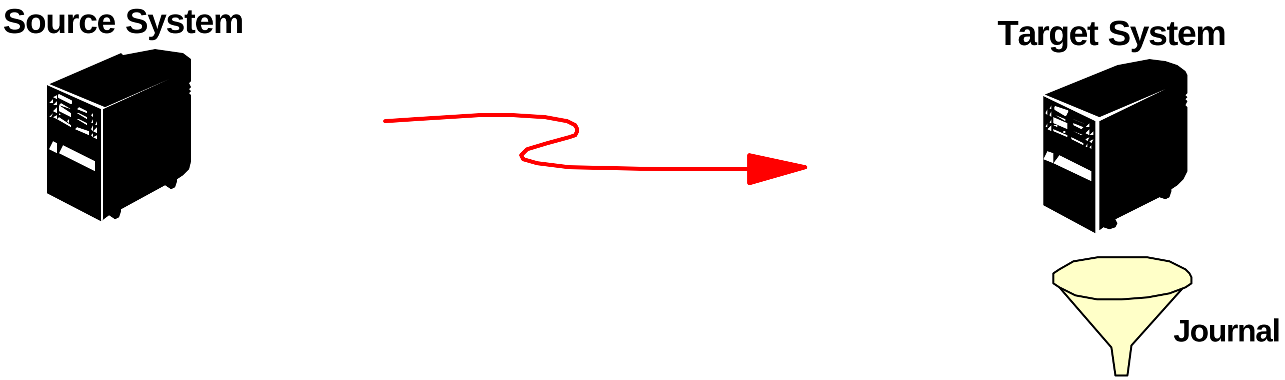
The longer they linger, the more the buffer fills and thus the fewer total disk writes on behalf of journal entries ensues -- and that's the primary reason the performance improves so dramatically.

When is it probably OK to let such changes linger longer in main memory ?
Ans: Probably during the execution of a batch job.

Performance Benefits of Journal Caching

(For Target Machine's Keep-up Mode)

	Keep up rate on Target machine
W/o Caching	600,000 transactions/Hr
With Caching on target	2,400,000 transactions/Hr



Notes:

While Journal caching is an effective tool in many scenarios on a production machine, the same journal caching behavior can also be mighty helpful in most instances on a **target** machine in an HA 24x7 environment.

If you employ HA Vendor software and would like to help the HA BP "apply/replay" jobs keep up on the target machine when you've got a journal intensive batch job running on the source machine, consider installing the Journal Caching on the target side as well.

Journal Overhead guidelines/trends

Normal Experience

batch runs made on an 840					
	extra time	extra cpu	cpu/JOE (us)	cpu/bundle (us)	extra writes
No Journaling	100%	100%			100%
Journaling with defaults	133%	102%	9.27	22.78	367%
user asp, RAID 5 10k rpm	130%	104%	15.06	37.08	367%
user asp, RAID 5 7200 rpm	133%	104%	14.76	36.75	364%
disk level mirroring	154%	108%	29.96	74.38	648%
*MAXOPT2	130%	104%	15.06	37.02	371%
*AFTER and omit *OPNCLO	131%	103%	17.26	29.09	371%
omit *OPNCLO	130%	103%	10.97	23.51	364%
RMVINTENT	133%	104%	15.36	37.70	366%
MINENTDTA	130%	104%	14.41	35.37	369%
*MINFIXLEN	133%	105%	17.61	43.17	369%
all options	128%	103%	22.30	30.65	366%
w/PRPQ	100%	99%			83%
w/PRPQ all options	96%	100%			80%

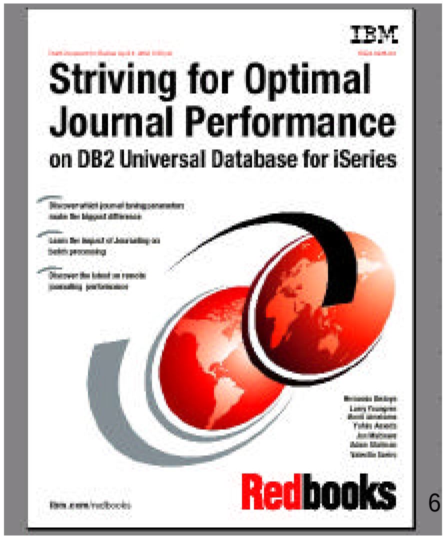
Opportunity

Appendix:
Summary of additional goodies found in the Redbook

Also see:

Jrn Performance articles

Jan & Feb issues: *iSeries News*



IBM

Striving for Optimal Journal Performance

on DB2 Universal Database for iSeries

- Discover which journal tuning parameters make the biggest difference
- Learn the impact of Journaling on batch processing
- Discover the latest on remote journaling performance

Redbooks

ibm.com/redbooks

Summary of Best Performance Practices

1. Chose wisely between mirrored or RAID protected disk drives

2. Configure up to 15 arms per user ASP
   - **The more arms the better** for journal intensive environments
   - Increases max bandwidth the journal receiver can support

3. Use modern/fast disk arms and an IOP with wider write cache
   - Model 2748 has 26 Meg vs. 4 Meg for older models

4. Spread User ASP across multiple IOPs
   - Helps reduce risk of IOP write cache overflow
   - Unless you have model 2748 IOPs or newer, configure no more than 5 busy Journal arms per IOP

(Continuation) of Best Performance Practices

5. **Issue OVRDBF** for batch (helps adds, not updates)
   - SEQONLY (*YES)
   - Use a separate ODP (view) to perform the adds
   - Specify NBRRCDS - - use large enough value
     - Make buffer size approach 128K

6. If your batch job performs update operations (not pure adds)
   - Issue commit verb every 1,000 batch updates (or 128k bytes)
   - Alternatively, surround batch job with single commit cycle
   - or... employ Batch Journal Caching PRPQ

7. Keep PF open for update in a secondary job
   - Especially if your job often opens/closes same file repeatedly

8. Don't journal nonessential files

9. Suppress open/close journal entries
   - STRJRNPF . . . OMTJRNE(*OPNCLO)

(Continuation) of Best Performance Practices

10. Employ CHGJRN . . . RcvSizOpt(*RmvIntEnt)
    - And add extra disk arms to the user ASP

11. Employ CHGJRN . . . RcvSizOpt(*MinFixLen)
    - Practical only if you don't need the extra information

12. Employ CHGJRN . . . MNGRCV(*SYSTEM)
    - Speeds up subsequent CHGJRN operations

13. Place no more than one active journal receiver per user ASP

14. Split long running batch job into parallel streams

15. Use SETOBJACC on both source and Target machines
    - Keeps modest sized keyed logical files resident

16. Don't save a journal receiver while still attached and filling
    - (Use the save/restore omit support: OMITOBJ parameter)

(Continuation) of Best Performance Practices

17. Capitalize on SMP feature to speed up Access Path maintenance

- **Use job-specific scoping via CHGQRYA**
  - **Most effective for batch jobs adding records to a physical file covered by dozens of access paths**
  - **Yields parallel index maintenance for blocked "Adds"**
  - **Speeds up "Adds" by servicing Access Path page faults in parallel**
  - **Some customers have seen up to a 30% performance improvement**
- **Only helps in concert with SEQONLY(*YES)**
  - **CHGQRYA . . . DEGREE(*OPTIMIZE)**
  - **CHGQRYA . . . DEGREE(*MAX)**

18. Consider increase of SMAPP recovery value on Target machine

- **In a High Availability environment**
  - **Keep it high until role swap ensues**
  - **Helps reduce overhead incurred by the apply jobs**

(Continuation) of Best Performance Practices

19. Start explicit journaling (STRJRNAP) on your largest access paths
    - **Especially if you've not been journaling the underlying PF**
    - **If you don't journal it, SMAPP will**
    - **Gives you control over placement of the journal receiver employed for SMAPP entries**
    - **Reduces disk arm contention and arm skew in the system ASP**
    - **Reduces CPU overhead by background SMAPP "Tuning" tasks**

20. Set Journal Receiver threshold appropriately

21. Pre-allocate space via CHGPF ALLOCATE(*YES)
    - **Reduces idle time and seize conflicts**
    - **Useful if your batch job creates/populates a new file**

22. Reduce contention within popular Keyed-logicals
    - **CHGLF ACCPTHSIZ(*MAX1TB) instead of *MAX4GB**
    - **Especially if your added key values have high locality of reference**

(Continuation) of Best Performance Practices

- 23. Employ extra-wide Stream IO for the files you read
  - **CHGQRYA DEGREE(*IO) or DEGREE(*MAX)**
  - **Then either:**
    - **A) Read via OPNQRYF**
    - **B) Override to employ an SQL view**

- 24. CALL QDBENCWT '1'
  - **and then IPL to enable "holey" blocked adds**
    - **Reduces contention among concurrent batch jobs performing SEQONLY blocked "Adds" to the same PF**
    - **Helpful if you've broken up batch job into multiple threads**
    - **Also requires REUSEDLT(*YES) on CHGPF**

- 25. Order and install PRPQ 5799-BJC (Aggressive Batch Journal Caching)
  - **Speeds up batch jobs performing update operations**
  - **Is less helpful if you employ commitment control**

- 26. **Always allocate space for VarChar columns**
  - **Without allocated space, Journal caching is less effective**

(Continuation) of Best Performance Practices

- 27. If you run purge programs to toss outdated records from your database files and empty out any of the history or work files, use the native **CLRPFM** rather than the SQL equivalent: **Delete ***
  - **CLRPFM emits only <u>one</u> journal entry and is quick**
  - **DELETE * emits a separate journal entry for <u>each</u> row discarded**

Trademarks and Disclaimers

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

AS/400
AS/400e
e (logo) business
IBM
IBM(logo)
iSeries
OS/400

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information in this presentation concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information in this presentation addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.