大規模DBサーバへのLinux適用

～Kernel2.6の実力を探る～

2005.11.11

NTTコムウェア株式会社
オープンソースソフトウェア推進部
野呂 昌哉

はじめに

Linux Kernel2.6の登場により、エンタープライズ分野における大規模DBサーバへのLinux適用の加速が期待されている。

今回はDB性能ベンチマークテスト結果をまじえ、Linuxの適用範囲が従来のWeb/APサーバ系フロントエンドシステムからDBサーバ系バックエンドシステムへ拡大可能であることを説明する。

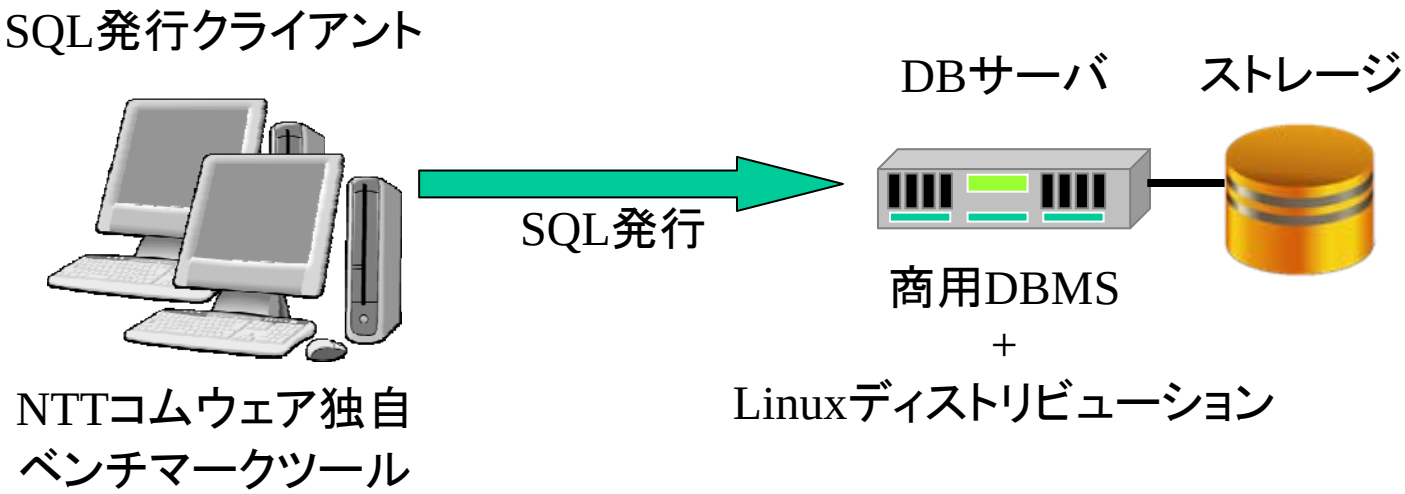
アジェンダ

- DB性能ベンチマーク
- IA32サーバ+Kernel2.4/2.6のDB性能
- IA64サーバ+Kernel2.6のDB性能
- 考察
- まとめ

DB性能ベンチマーク

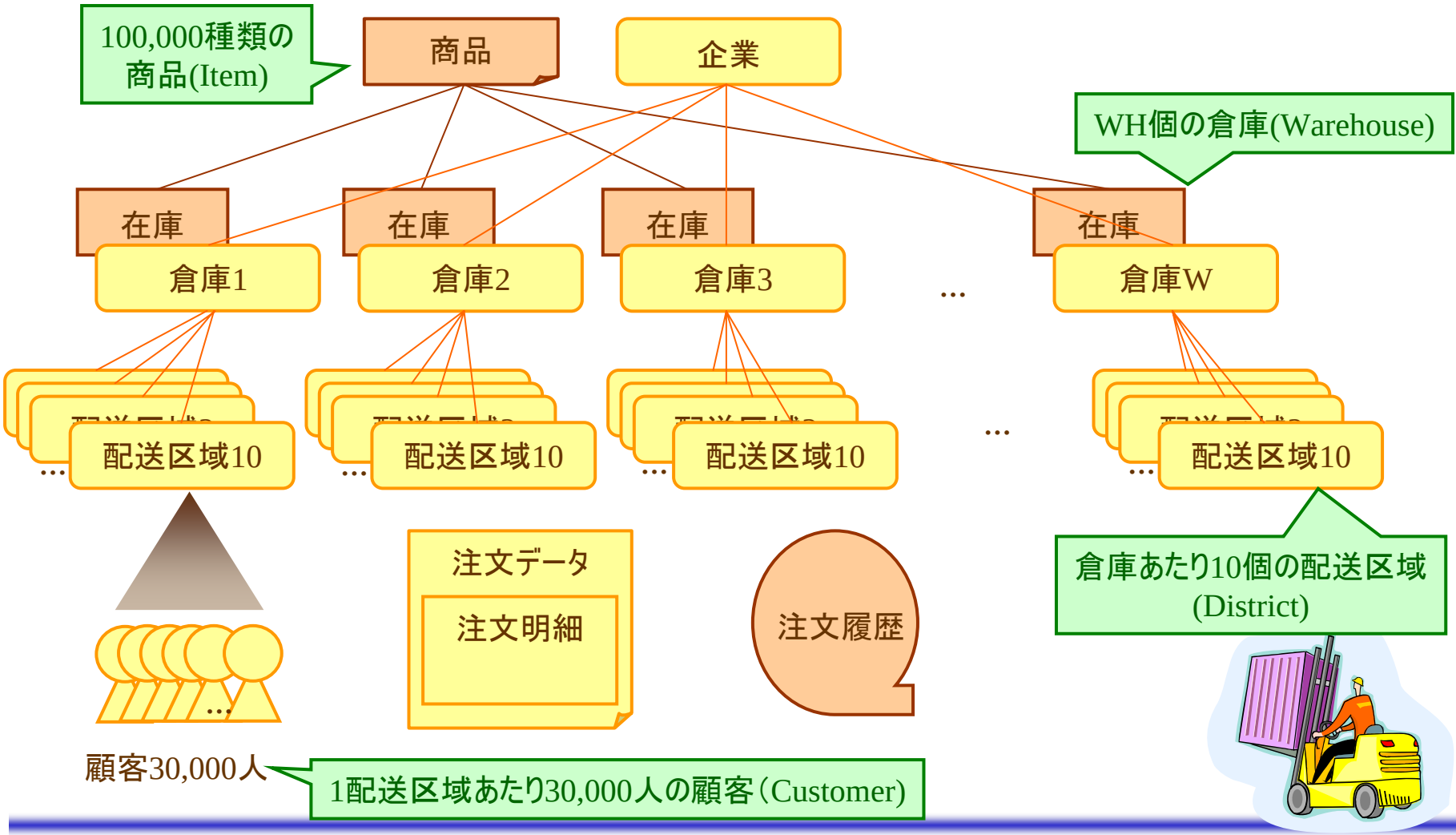
ベンチマークテスト概要

■ DBサーバの単位時間あたりのトランザクション数(スループット)を測定



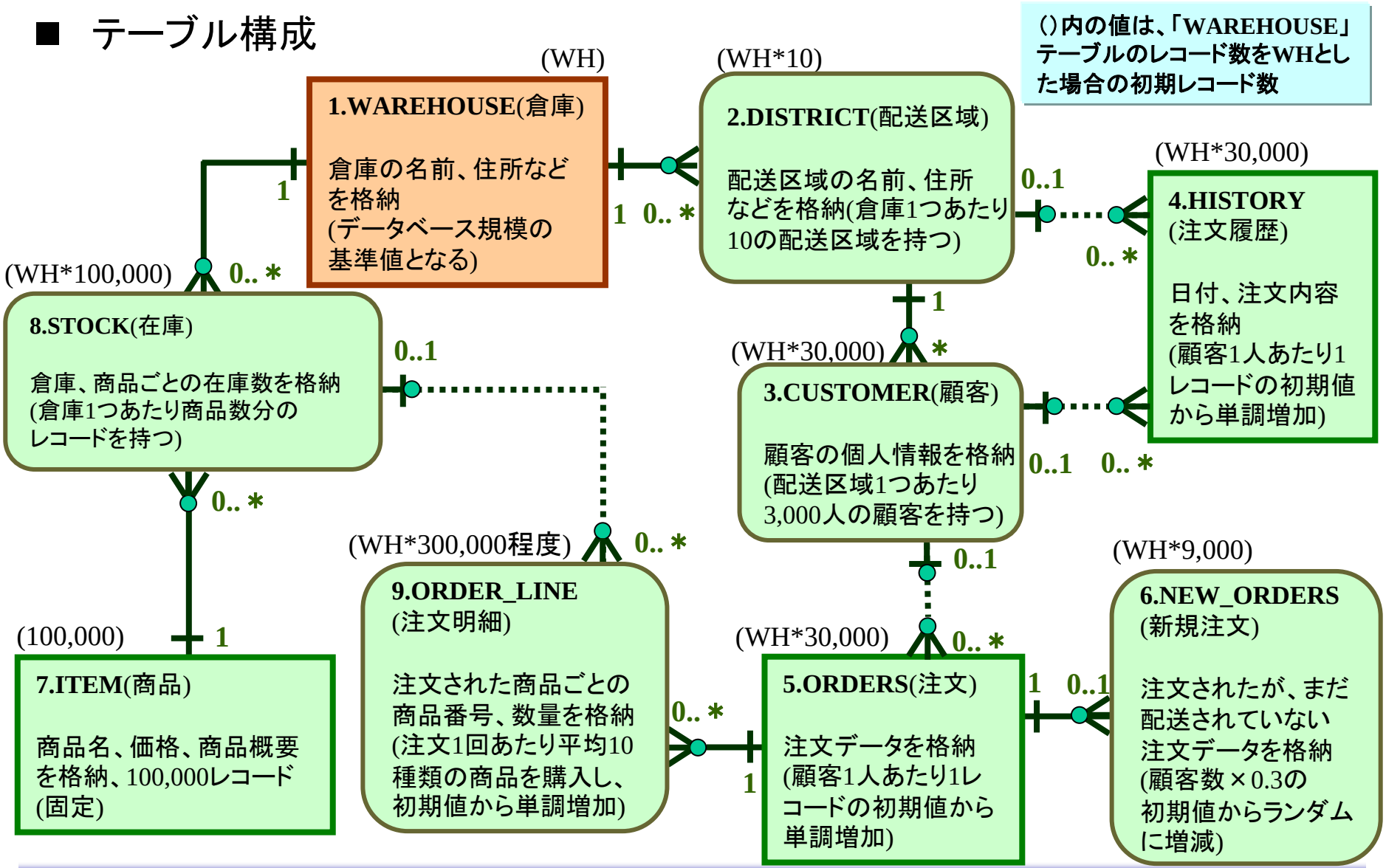
ベンチマークテストモデル

■ 典型的な卸売り業を想定したテスト



DBスキーマ

■ テーブル構成



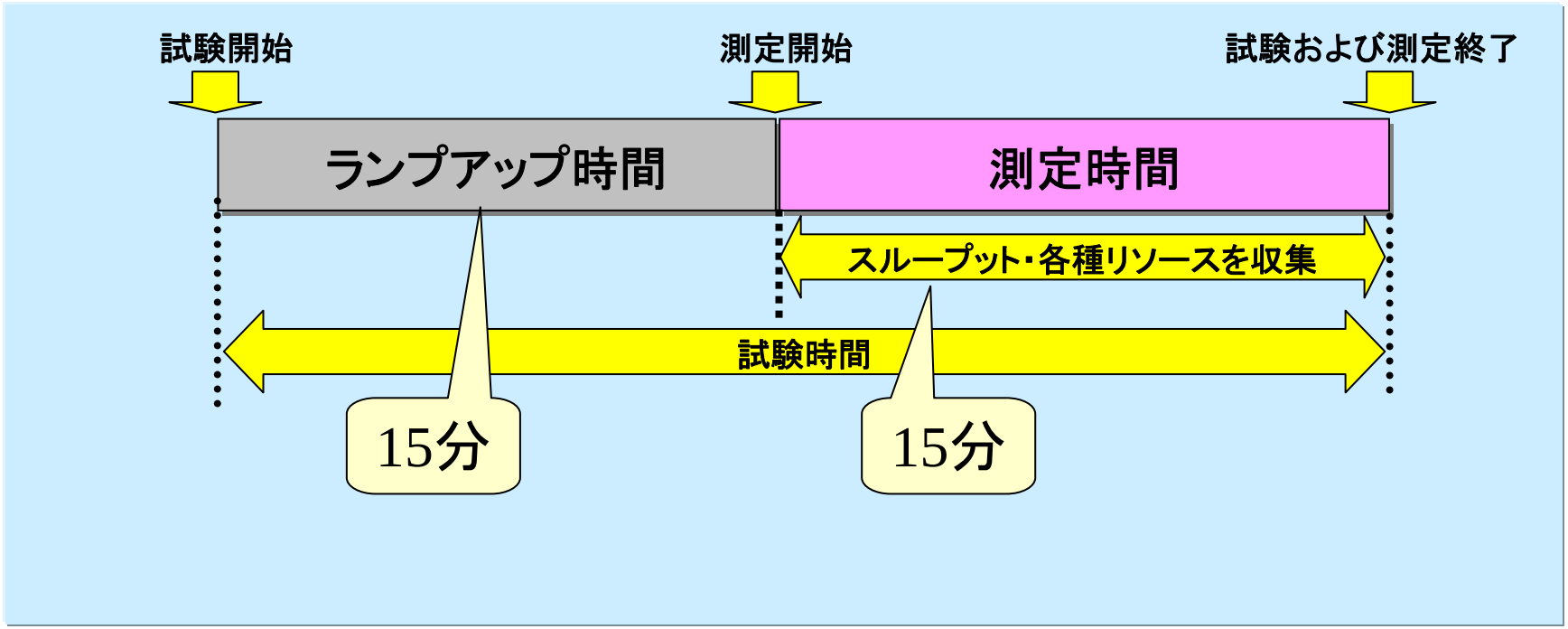
発行SQL

■ スループット＝New-Orderの1分あたりの件数(Tpm)

トランザクション	SELECT	INSERT	UPDATE	DELETE	小計	実行比率	レスポンスタイム
New-Order	22回	12回	11回	—	45回	(45%以下)	90%が5秒以内
Payment	3回	1回	3回	—	7回	43%以上	90%が5秒以内
Order-Status	3回	—	—	—	3回	4%以上	90%が5秒以内
Delivery	30回	—	30回	10回	70回	4%以上	90%が5秒以内
Stock-Level	202回	—	—	—	202回	4%以上	90%が20秒以内

測定方法

■ 安定するまでランプアップを確保



測定ポイント

■ リソース、DBMS統計情報、ストレージ性能を収集し、ボトルネックを解析

データ収集項目	クライアント	DBサーバ	ストレージ	収集用途
sarコマンド	○	○	―	CPU使用率/ディスク使用状況/メモリ使用状況などを確認する。
vmstatコマンド	○	○	―	
iostatコマンド	○	○	―	
freeコマンド	○	○	―	
psコマンド	○	○	―	検証目的以外のプロセスでCPUに負荷をかけているプロセスがないか確認する。
DBMS統計情報	―	○	―	DBMSの統計情報を収集し、DBMSの状態を確認する。
ストレージ	―	―	○	ストレージ側からみたI/Oの状態を確認する。

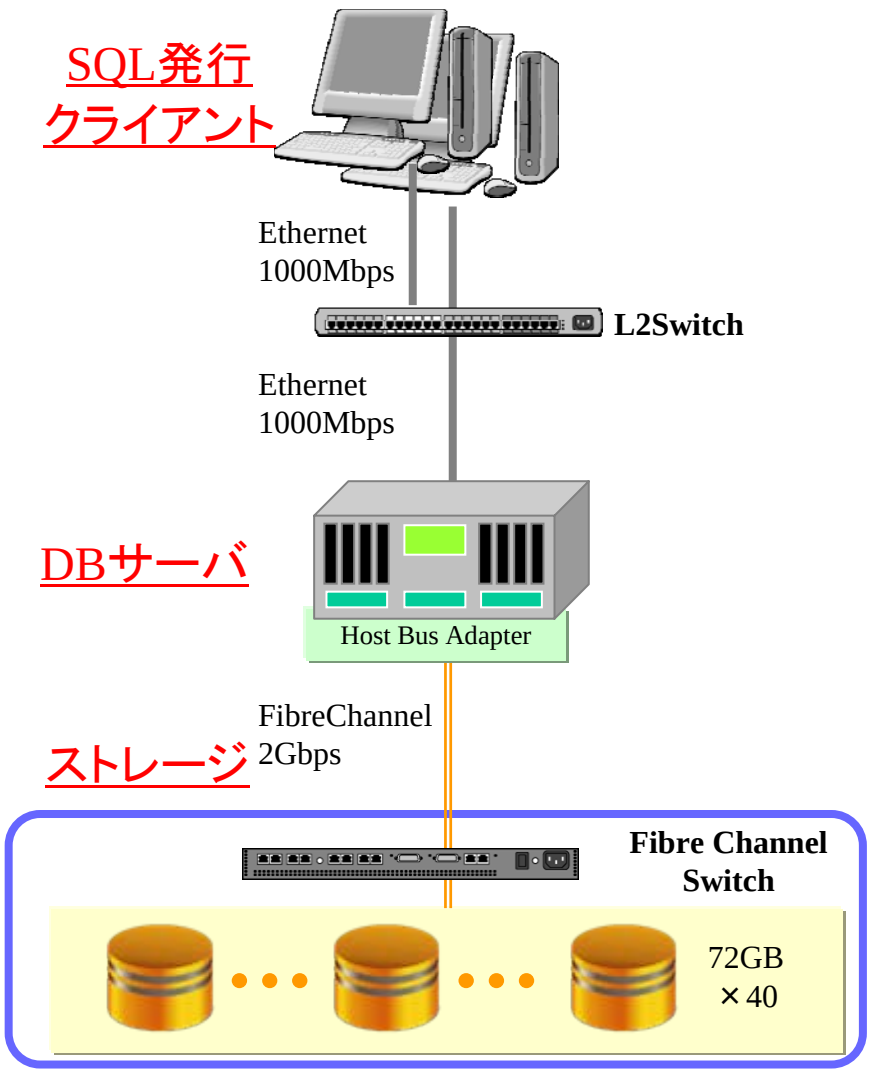
（凡例） ○：収集する　―：収集しない

IA32サーバ+Kernel2.4/2.6のDB性能

目的

- Kernel2.4→2.6に伴うDB性能向上度合いを確認する
  - ハードウェア、DBMSは同一条件とし、LinuxディストリビューションのみKernel2.4/2.6対応の2種類を使用
  - DBサーバ以外の部分でボトルネックにならないように高速なストレージ、複数台の高速クライアントを使用
- Kernel2.6のI/Oスケジューラの違いによるDB性能差を確認する
  - Kernel2.6で選択可能となった4種類のelevatorアルゴリズムを変更し、性能に与える影響を確認

試験環境



IA32サーバ×2	
CPU	Intel Xeon3.2GHz×2
Memory	5GB
NIC	Ethernet 1000Mbps Full Duplex
Linux OS	Kernel 2.4.21-32.0.1 (smp)

L2Switch	
Speed	Ethernet 1000Mbps

IA32サーバ	
CPU	Intel XeonMP3.0GHz×4
Memory	16GB
NIC	Ethernet 1000Mbps Full Duplex
HBA	2Gbps Full Duplex
Linux OS	Kernel 2.4.21-32.0.1 / 2.6.9-11 (smp)
DBMS Setting	AsyncI/O & DirectI/O

ストレージ	
HDD	72GB(15000rpm) ×40
RAID	RAID1+0
FileSystem	ext3 (mount mode=ordered, noatime)

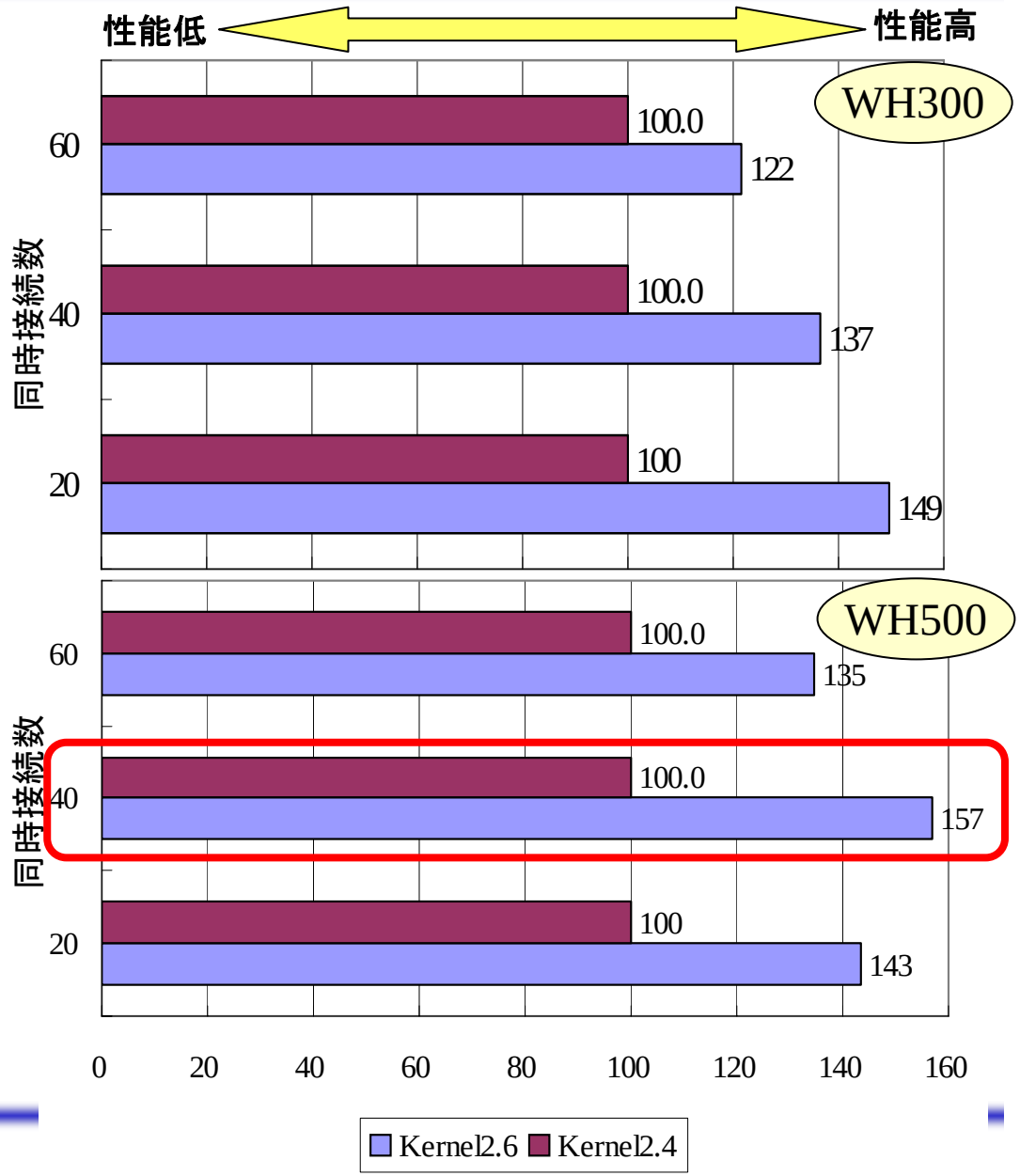
ー　Kernel2.4/2.6の比較　ー

Kernel2.4/2.6のDB性能比較

- Kernel2.4のスループットを100とした性能比

Kernel2.6の処理性能は
Kernel2.4の最大1.6倍

その要因は後ほど考察で. . .

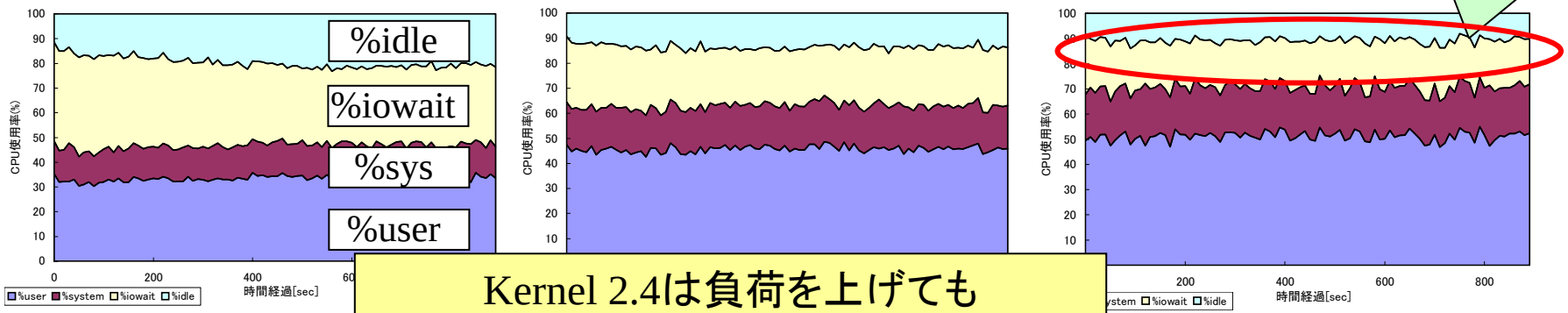


Kernel2.4/2.6のCPU使用率の比較

- CPU使用率時間推移をグラフ化
  - Kernel2.4では負荷(接続数)を増やしてもスループットは頭打ちでCPUを使い切れない
    →I/Oが先にボトルネックに！

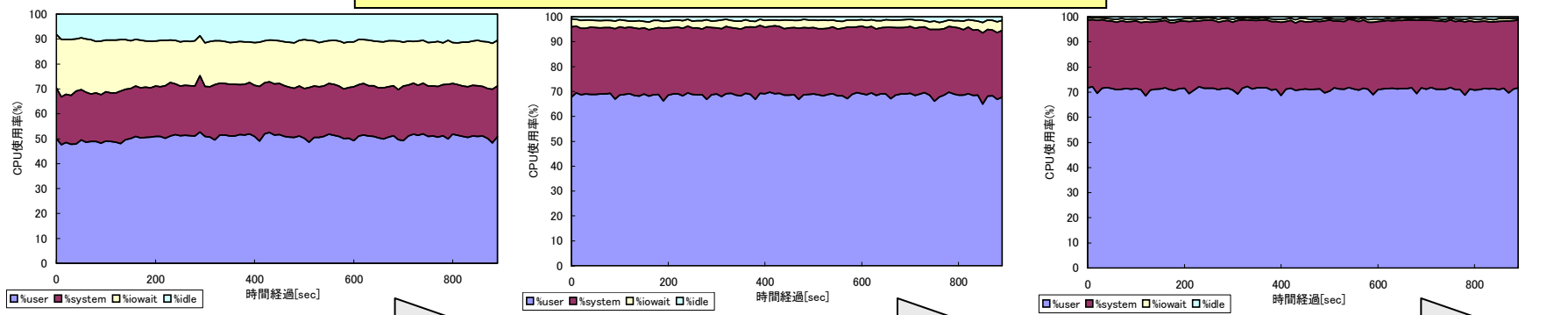
同時接続数20　　同時接続数40　　同時接続数60

Kernel2.4 (WH300)



Kernel 2.4は負荷を上げても
CPUリソースのiowait、idleが残る

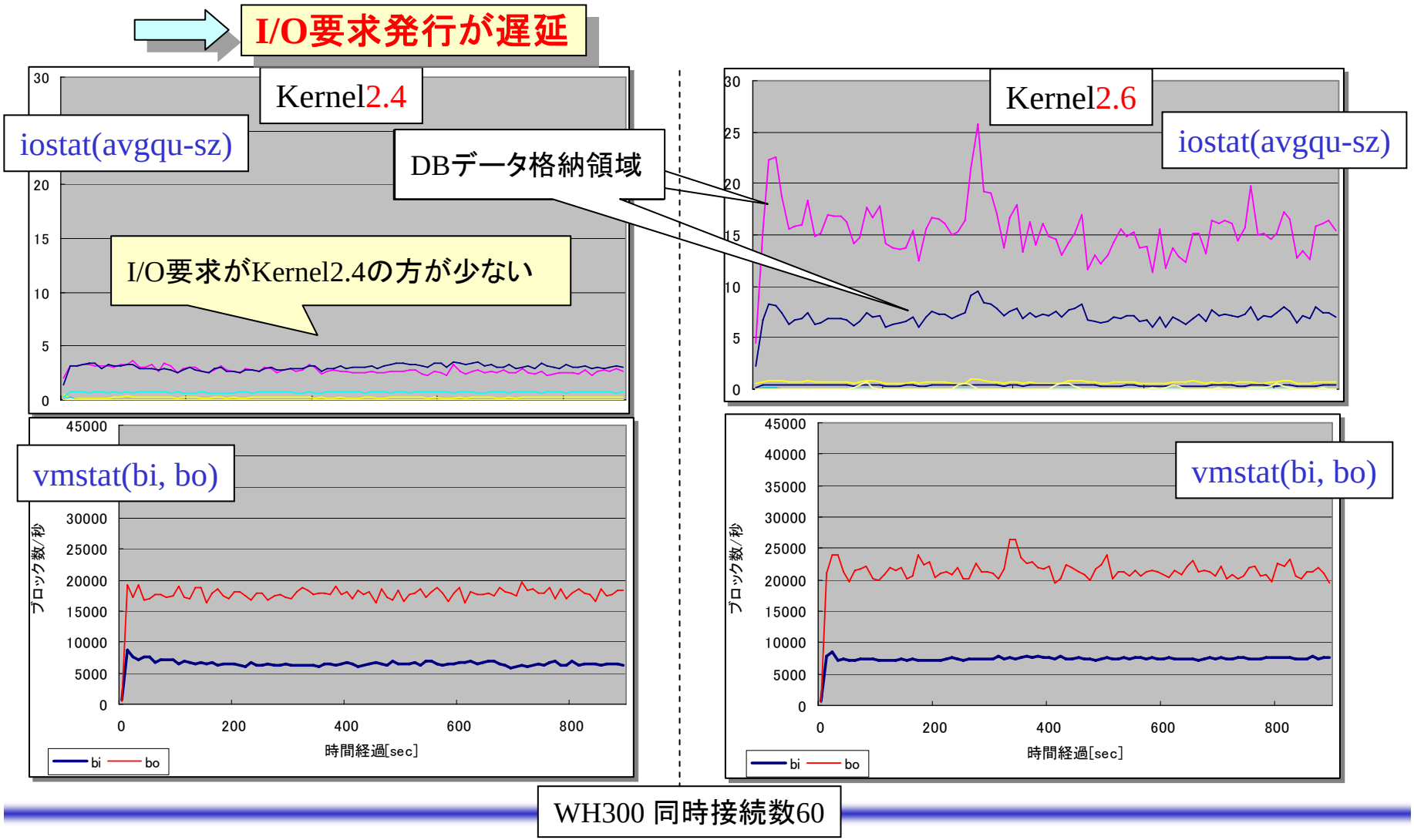
Kernel2.6 (WH300)



時間　　時間　　時間

I/Oの状態について(Kernel2.4/2.6)

- サーバのiostat
  - CPU使用率からはKernel2.4の方がI/O処理がボトルネックに見えるが実際には余裕の状態

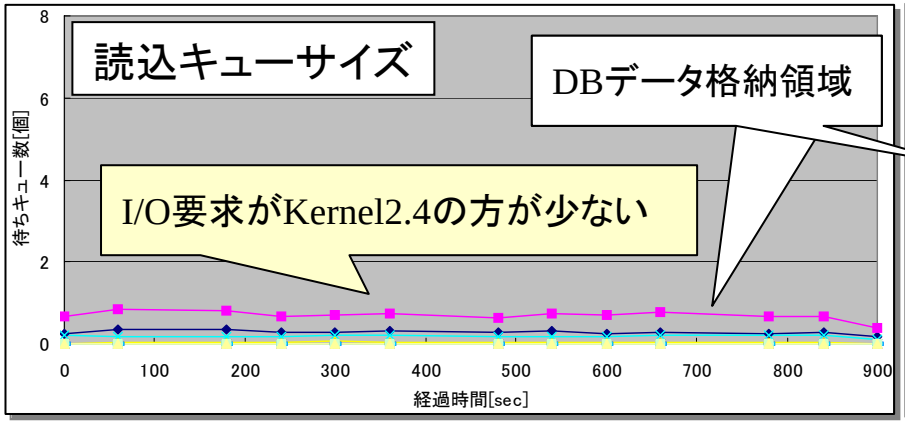


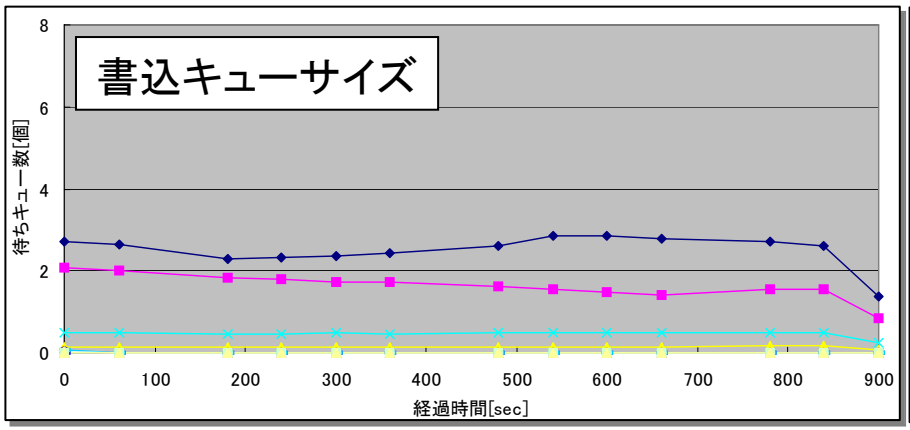
WH300 同時接続数60

I/Oの状態について(Kernel2.4/2.6)

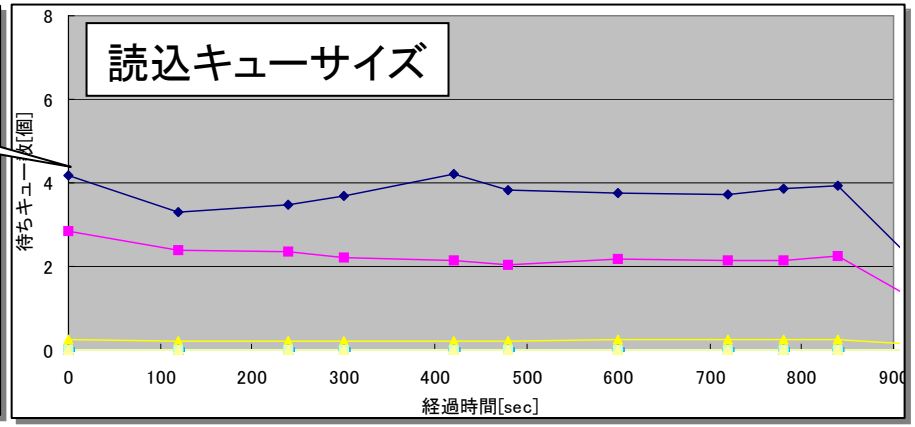
■ ストレージのI/Oキューの状態

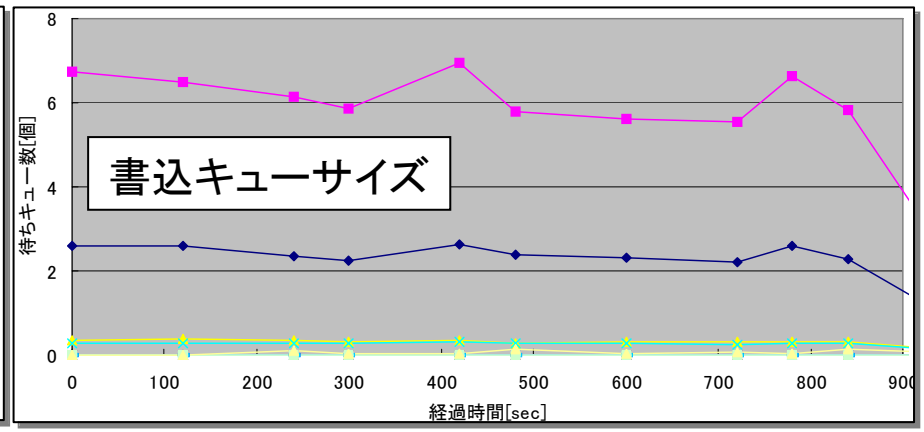
Kernel2.4





Kernel2.6





WH300 同時接続数60

ー　Kernel2.6のI/Oスケジューラ　ー

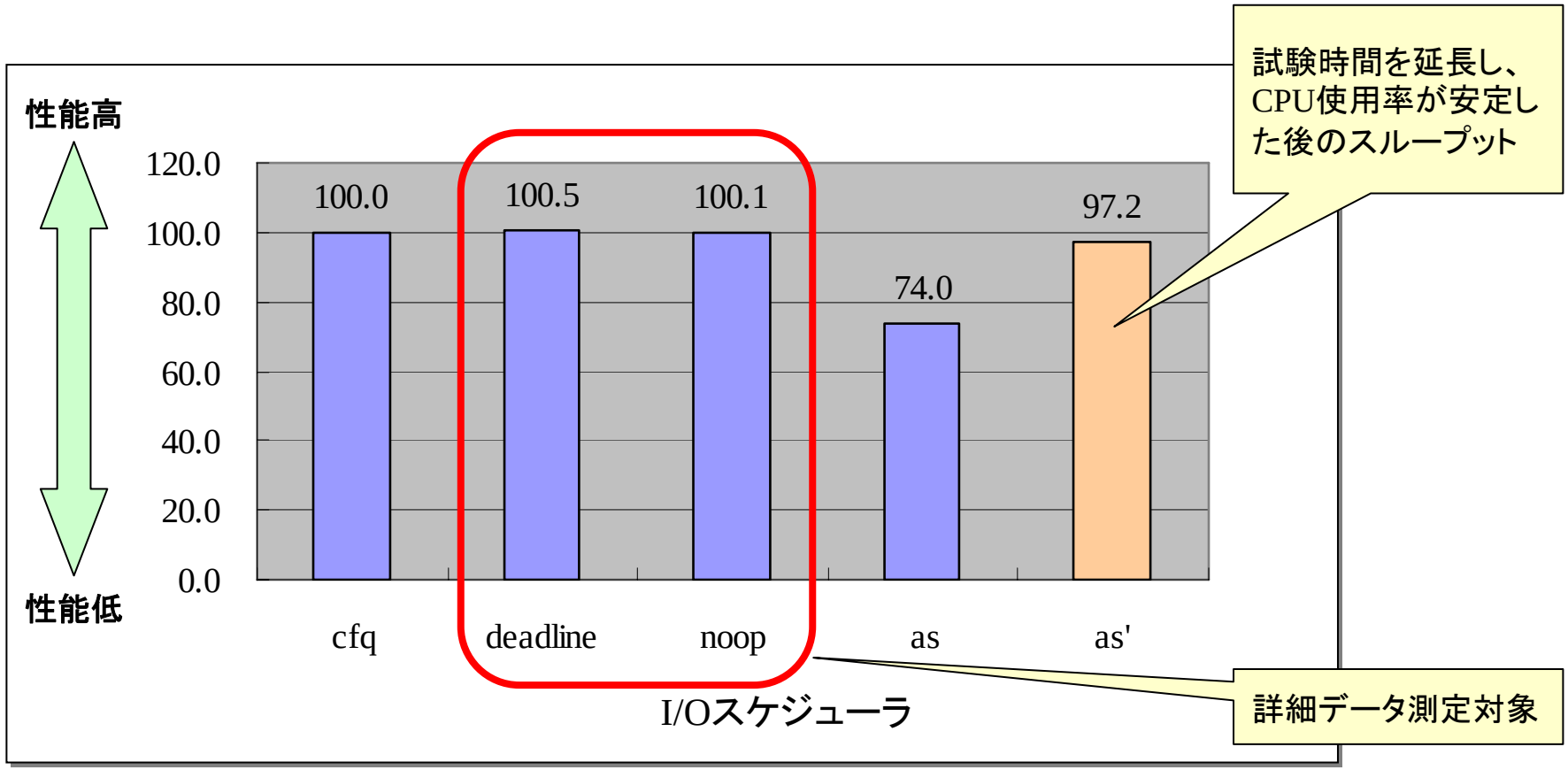
Kernel2.6のI/Oスケジューラ

■ Kernel2.6には、I/Oスケジューラ(elevatorアルゴリズム)が4種類ある

elevator	概要	適用範囲
cfq (Complete Fair Queuing)	I/O要求を各プロセス毎で時間に対して均一に割り当てる。	・中～大規模のマルチプロセッサシステム ・複数のLUNとI/Oコントローラ間に均衡なI/O性能を要求するシステム
deadline	I/O要求がディスパッチされる期限を設定して応答性を確保する。読み込みを優先する。	・DBMS ・リアルタイムシステム用途
noop (No Operation)	I/O要求の単純なマージ作業を行うだけでそのままディスパッチする。	・組み込み向け ・高性能なディスクや不規則にアクセス可能なメモリディスク
as (Anticipatory)	deadlineをベースにしており、遺伝的アルゴリズムを利用したマージや順序入れ替えのパターンを学習する。	・小規模あるいは遅いディスクサブシステムを持つシステム、例えばファイルサーバ

Kernel2.6のI/Oスケジューラ性能比較

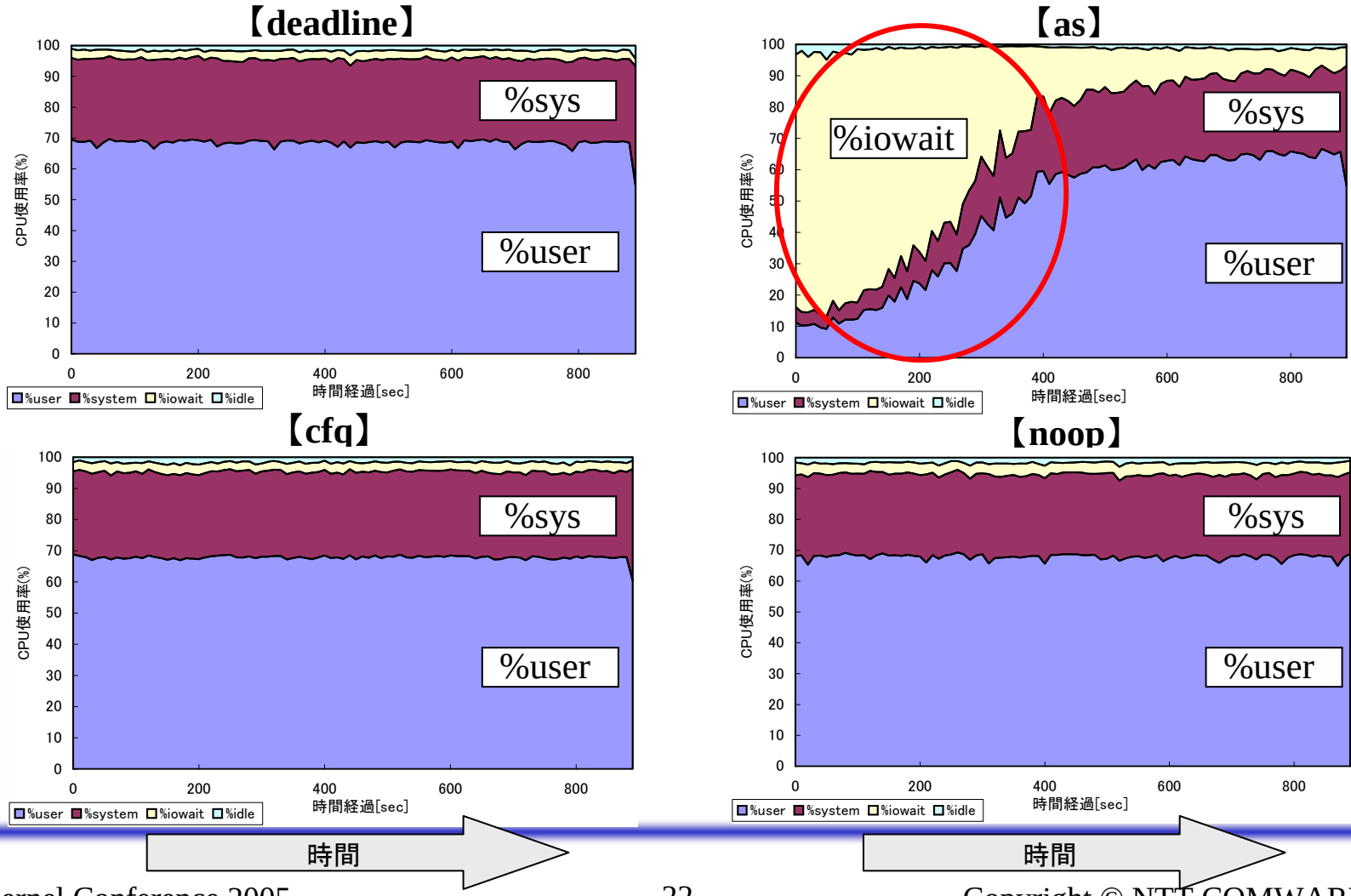
■ Kernel2.6のI/Oスケジューラ毎の性能比較(cfqのスループットを100とした)



※WH300、同時接続数40

I/Oスケジューラ毎のCPU使用率の比較

- CPU使用率時間推移をI/Oスケジューラ毎にグラフ化
  - asは測定開始当初は%userが低い＝スループットが劣る原因
    →測定時間を長く取れば、他と同程度のCPU使用率となり、スループットも追いつく

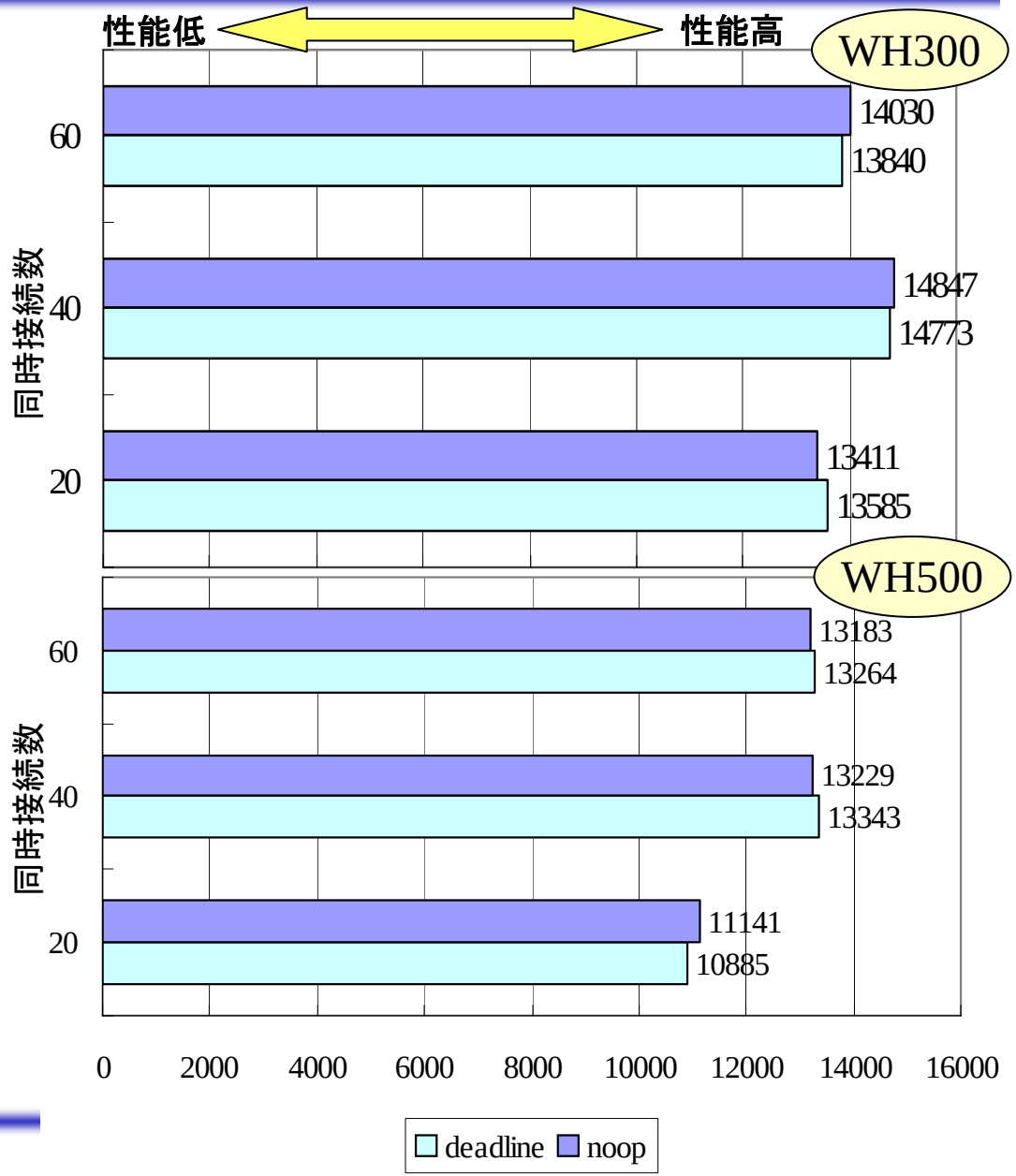


noop/deadlineのDB性能比較

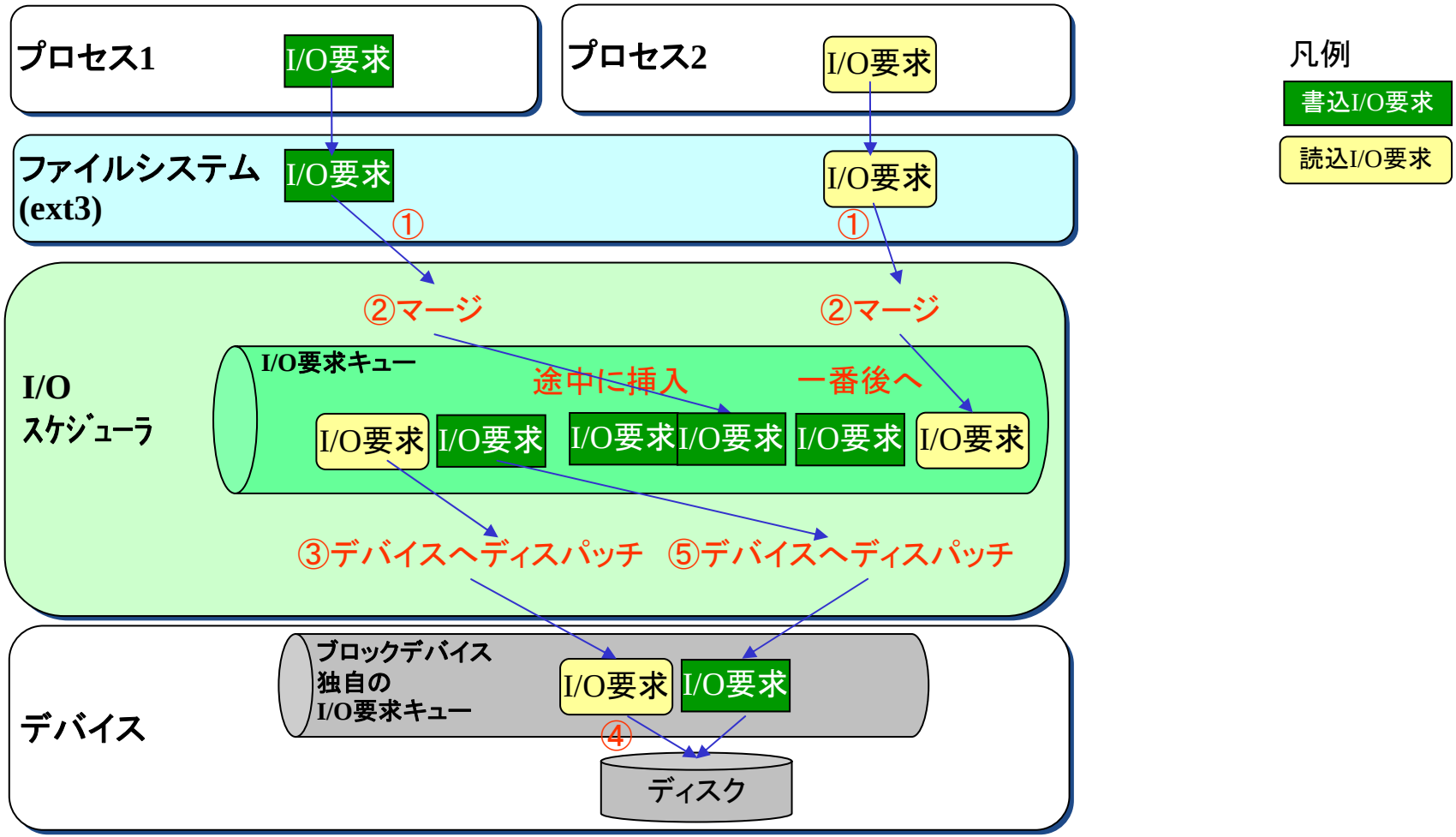
- noopとdeadlineの性能を測定

noop、deadlineの違いによる
性能差はほとんどない

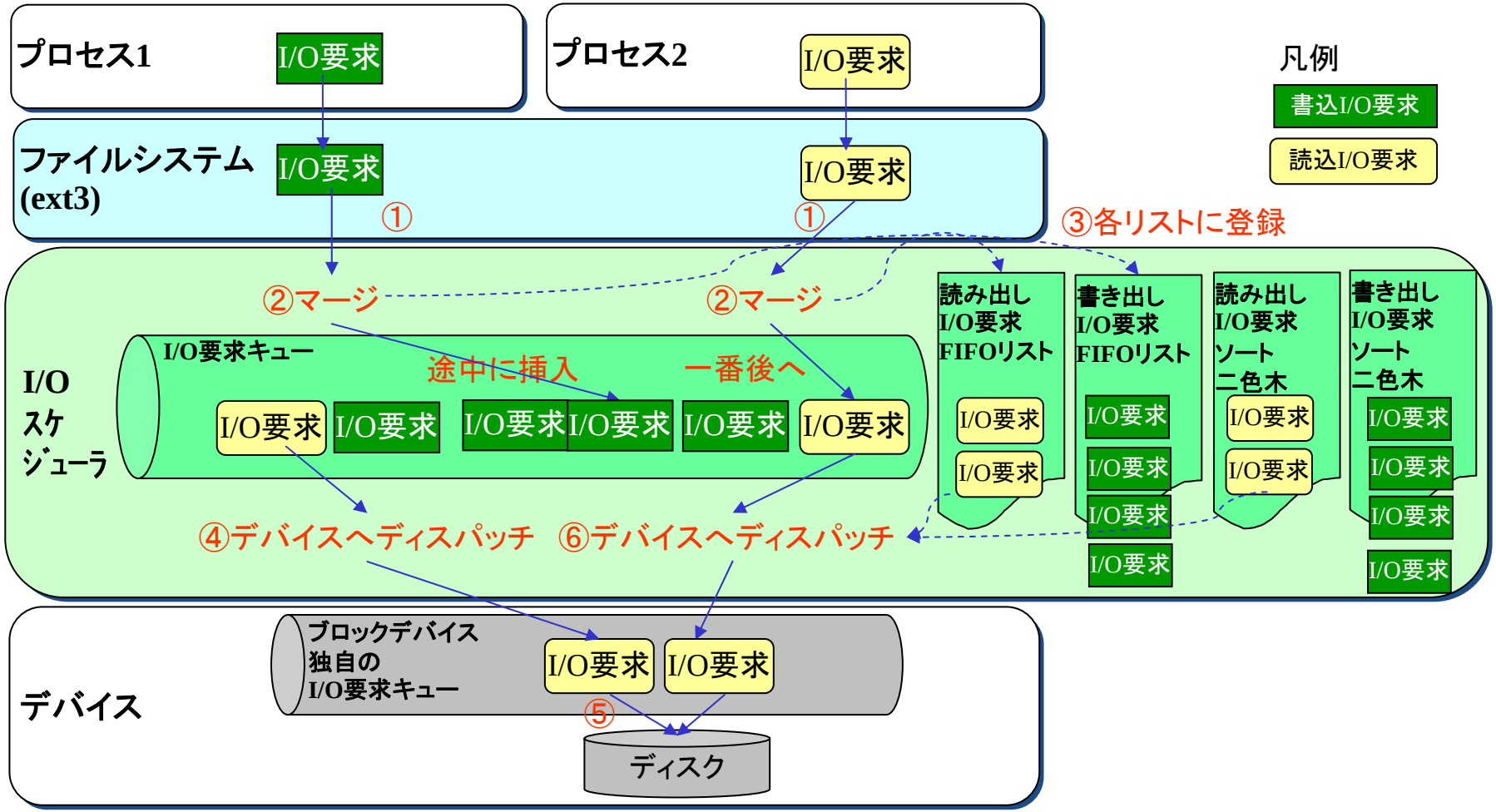
理由は...



■ noopスケジューリングの概要



■ deadlineスケジューリングの概要



まとめ

■ 検証結果より

- Kernel2.4→2.6の変更に伴い、I/Oデバイスを効率良く使うことができ、最大1.6倍の性能向上が得られた

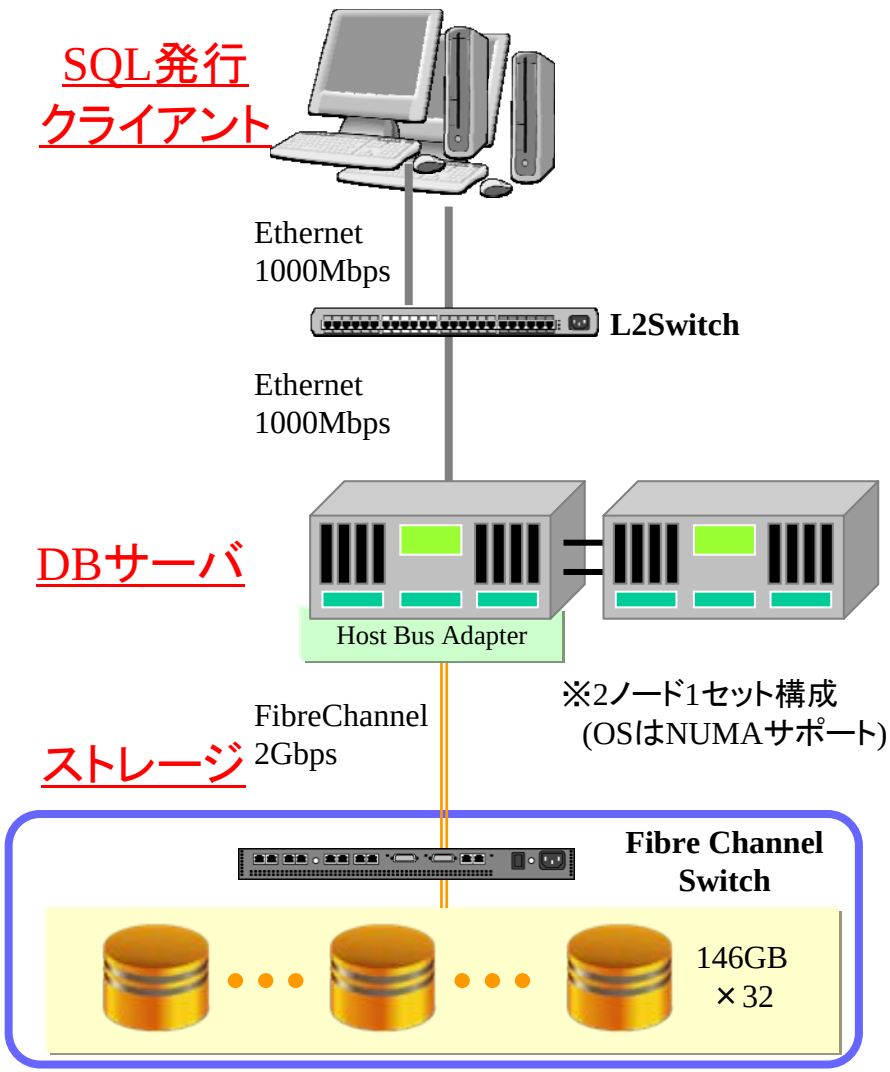
- LinuxのI/Oスケジューラはヘッドが1つの単純なディスクを想定しているため、I/O要求をセクタ順に並び替える必要のない高速・インテリジェントなストレージを用いる場合、I/Oスケジューラはnoop、deadlineのどちらを選択しても大差はない

IA64サーバ+Kernel2.6のDB性能

目的

- Kernel2.6のDBサーバ性能におけるCPUスケーラビリティを確認する
  - ハードウェア、DBMSは同一条件とし、CPU数のみ2, 4, 8に変更
  - DBサーバ以外の部分でボトルネックにならないように高速なストレージ、複数台の高速クライアントを使用

試験環境



IA32サーバ×2	
CPU	Intel Xeon3.4GHz×2
Memory	6GB
NIC	Ethernet 1000Mbps Full Duplex
Linux OS	Kernel 2.4.21-32 (smp)

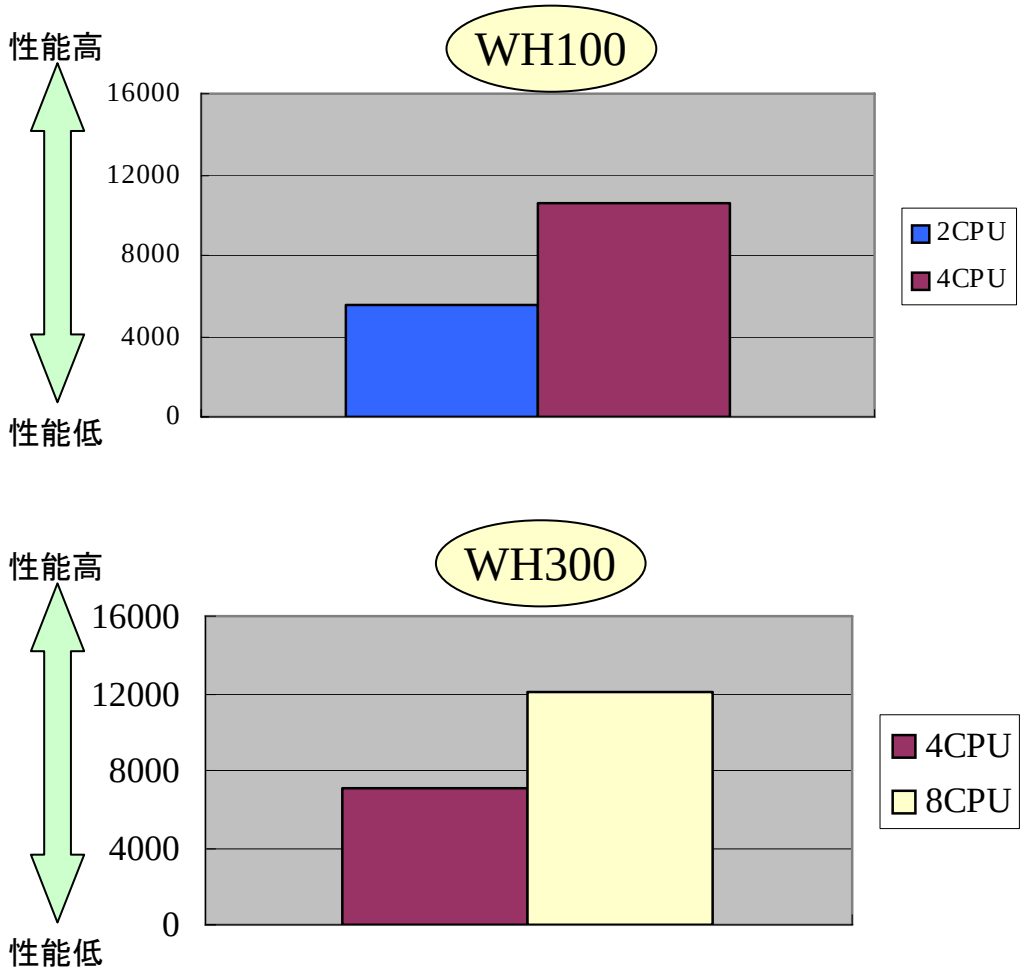
L2Switch	
Speed	Ethernet 1000Mbps

IA64サーバ	
CPU	Intel Itanium2 1.3GHz×8 (4+4CPU)
Memory	18GB (9+9GB)
NIC	Ethernet 1000Mbps Full Duplex
HBA	2Gbps Full Duplex
Linux OS	Kernel 2.6.9-11 (smp)
DBMS Setting	AsyncI/O & DirectI/O

ストレージ	
HDD	146GB(10000rpm) ×32
RAID	RAID0
FileSystem	ext3 (mount mode=ordered, noatime)

■ DB規模WH100/WH300のスループット

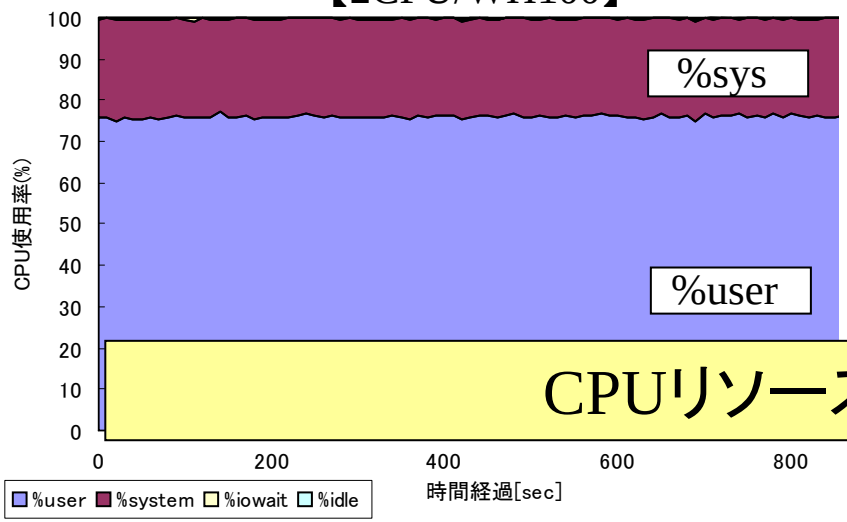
- 2CPU / 4CPUのスループット比 1.9倍
- 4CPU / 8CPUのスループット比 1.7倍



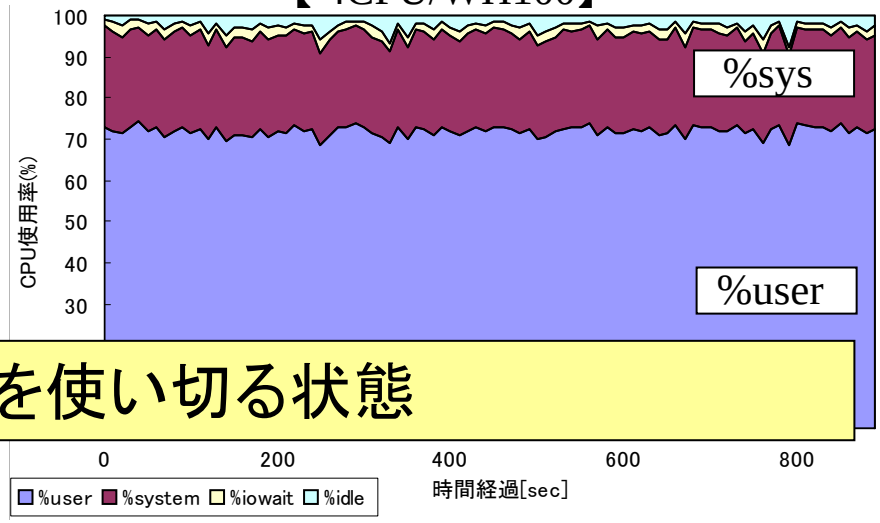
CPUリソースの使用状況

■ CPU使用率推移

【2CPU/WH100】

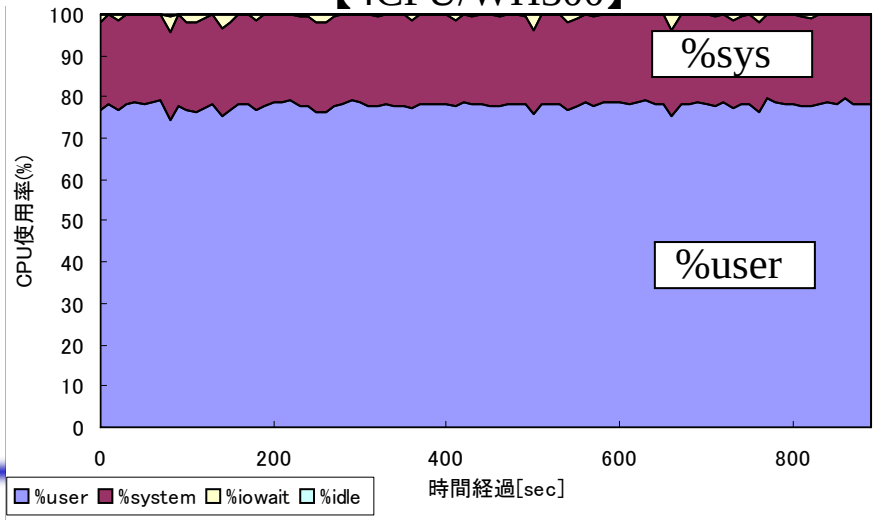


【 4CPU/WH100】

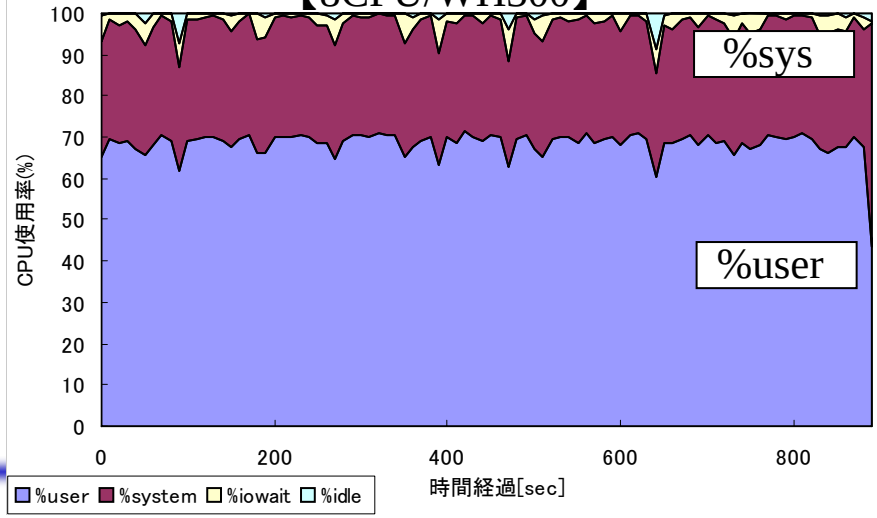


CPUリソースを使い切る状態

【4CPU/WH300】



【8CPU/WH300】



まとめ

- 検証結果より
  - Kernel2.6を使用することで8CPUまでスケーラビリティを確保できる

考察

なぜKernel2.6は2.4と比較してDB性能が向上したか?

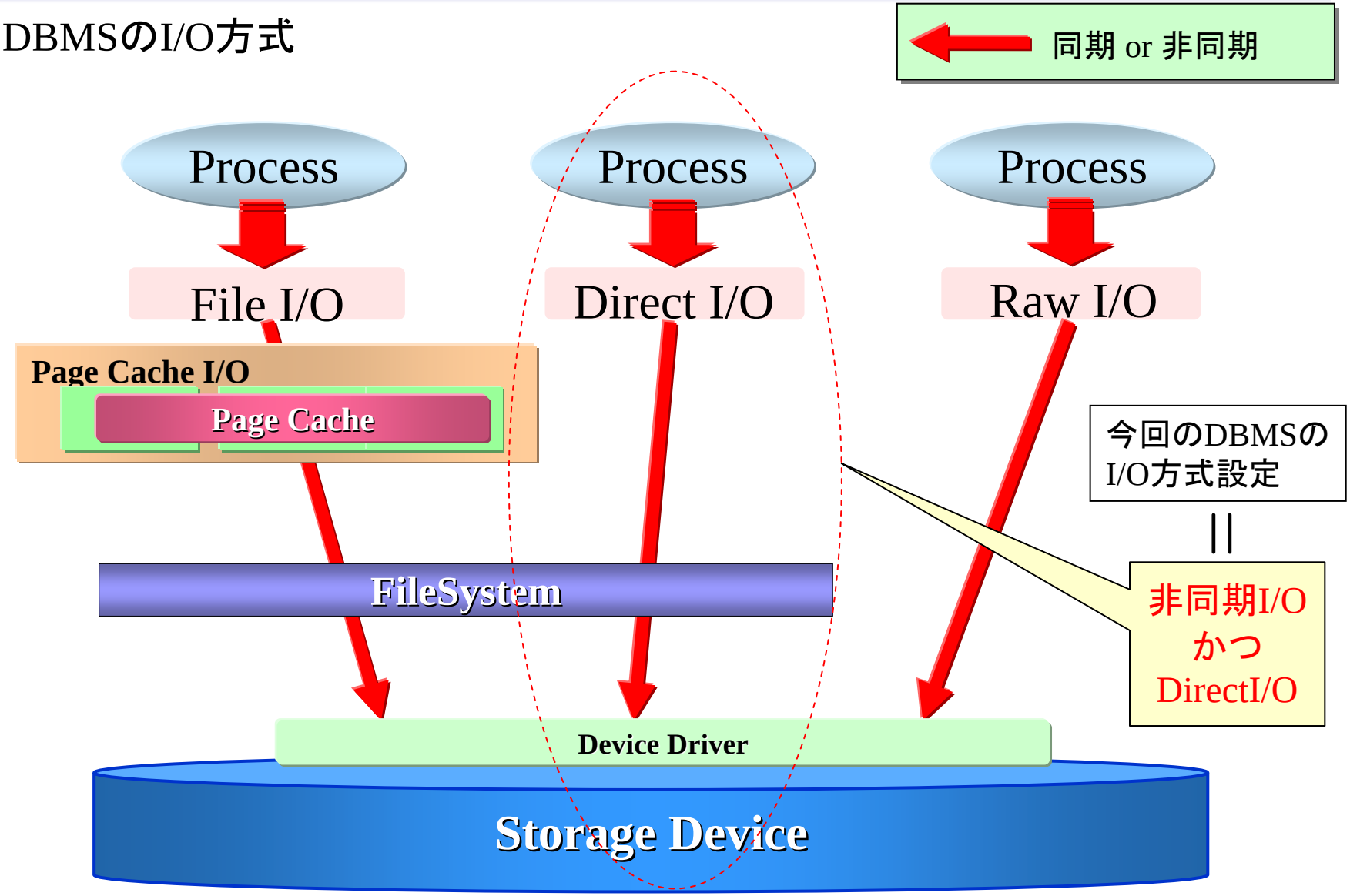
- Kernel2.4と比較して、Kernel2.6のDBサーバ性能が優れていた要因をKernelの観点から探る

  - ボトルネックにならないように高速なストレージを使用したにもかかわらず、Kernel2.4ディストリビューション環境下ではI/O要求発行が遅延してしまったのはなぜか？

  - Kernel2.6ディストリビューションではI/O処理に関して何が改善されたのか？

DBMSのI/O方式

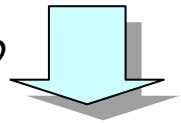
■ DBMSのI/O方式



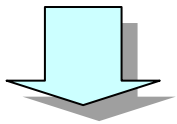
Kernel2.4/2.6の“非同期I/O+Direct I/O”の実装の違い

- Kernel2.4/2.6ディストリビューションの「非同期I/OとDirect I/O」の組み合わせの実装の違い

  - DBMSはKernel2.4/2.6でlibaioライブラリの同じ関数を使用
    - ファイルオープンモード　　: O_SYNC | O_DIRECT
    - システムコール　　: io_submit(), io_getevents()

まったく同じにみえるものの  *実は…*

  - 今回使用したKernel2.4ディストリビューションは“非同期I/OかつDirectI/O”の設定をしても、**DirectI/Oがサポートされていなかった**



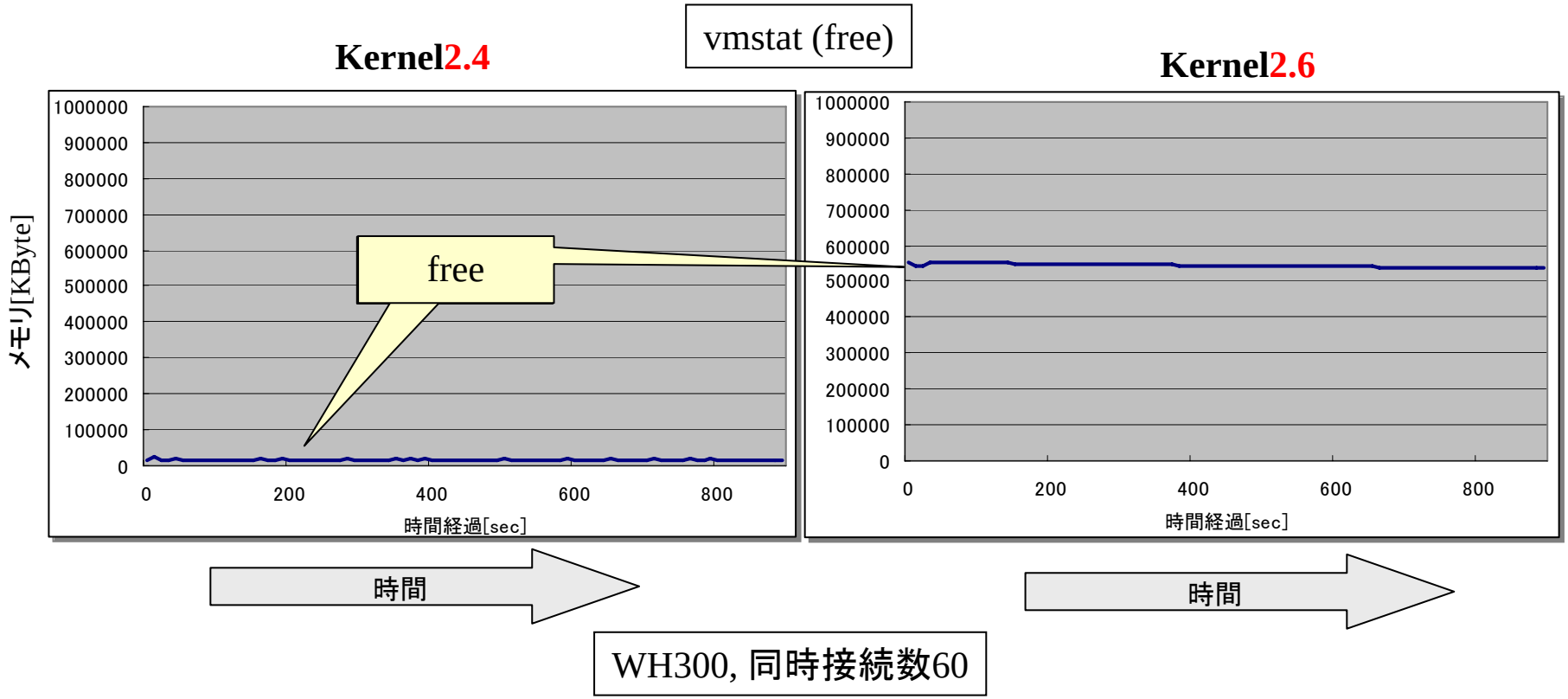
  - ページキャッシュを使用するため、メモリを大量消費

※メモリを大量消費すること自体は必ずしも性能劣化につながらない

メモリ使用量の違い

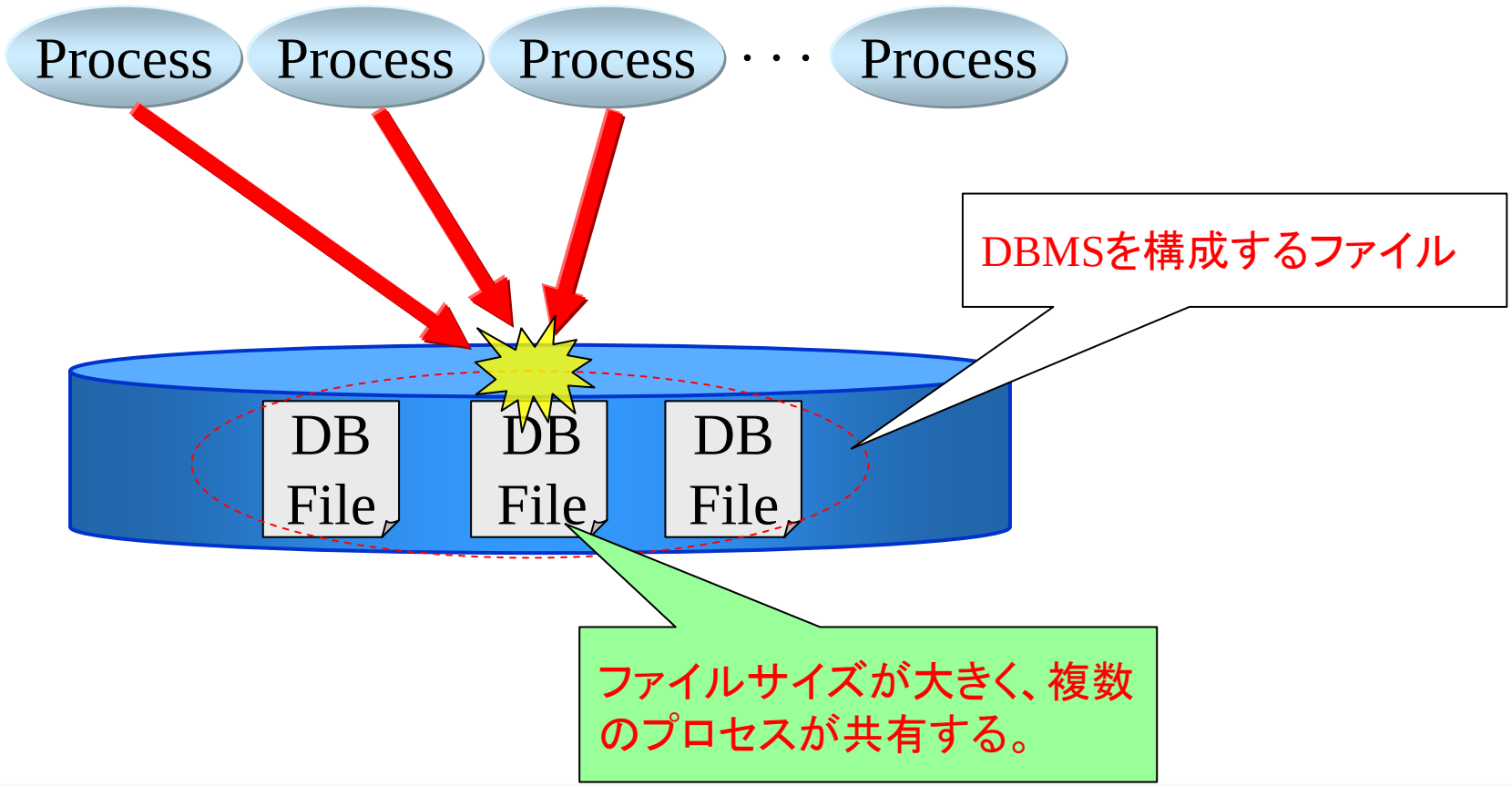
■ Kernel2.4/2.6ディストリビューションのメモリ使用量の違い

□ 今回使用したKernel2.4ディストリビューションはDirectI/Oが有効にならず、ページキャッシュに使用されるので**freeが少ない**



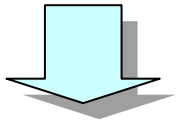
■ DBMSの特徴

- 今回のDBMSは1つの巨大なファイルを複数プロセスが共有する仕組み



メタデータを格納していたキャッシュの解放

- Kernel2.4でページキャッシュにメモリが消費される影響
  - メモリが大量消費され、freeがなくなる



  - **メタデータ**を格納していたページが解放される

※メタデータ＝実データのディスク上の位置やファイルサイズ、更新日時等の、ファイルに関するデータ

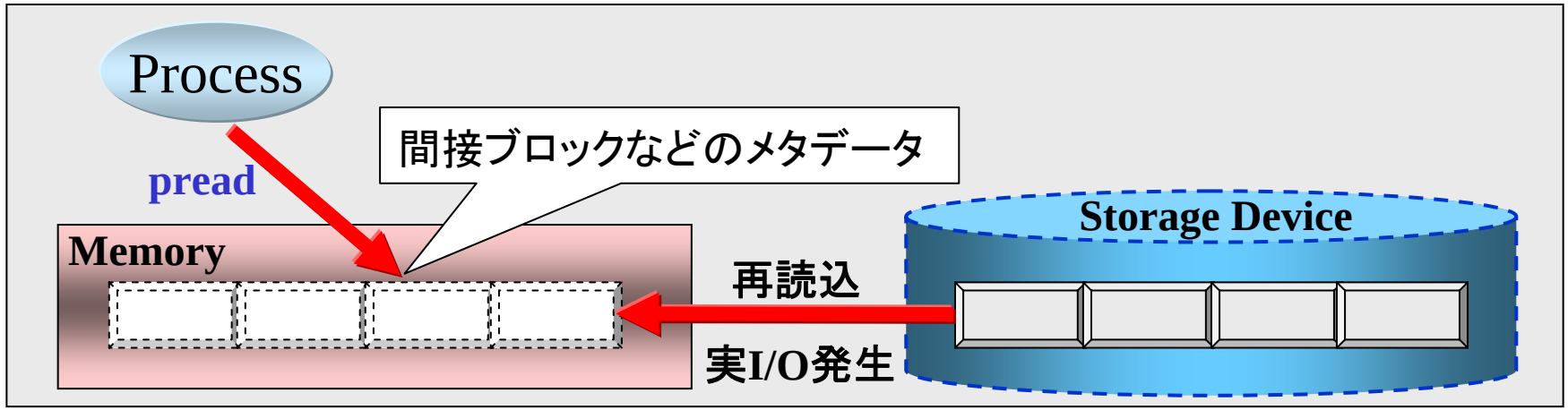
CPUリソースのiowaitが発生する要因(1)

■ メタデータが格納されていたページが解放される影響①

- 今回使用したDBMSは、“非同期I/O+DirectI/O”設定時にもサーバプロセスが同期I/Oとなる**preadシステムコールを同時に大量発行**していた

- メタデータがメモリ上にないため、メタデータの再読込I/O完了待ち合わせでプロセスがブロックされる

CPUリソースの%iowaitの増加



- 本事象はrawデバイス場合は発生しない。ただし、運用面を考慮するとrawデバイスは選択しづらい。

CPUリソースのidleが発生する要因(1)

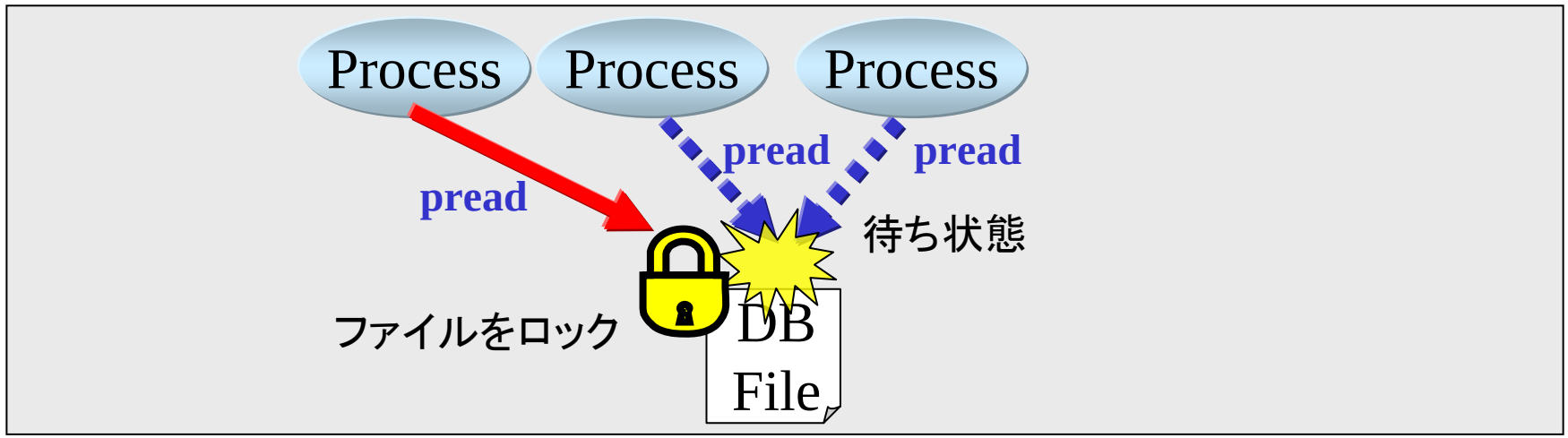
■ メタデータが格納されていたページが解放される影響②

- preadシステムコールはメタデータアクセス中、セマフォを獲得して**ファイルをロック**

今回のDBMSは同一ファイルを ⇒ *複数プロセスが共有*

- **他のプロセスが同一ファイルにI/O要求**しても、別のプロセスがメタデータの再読込処理をしている間、**ファイルがロックされているため処理が進まない**

⇒ CPUリソースの%idleの増加



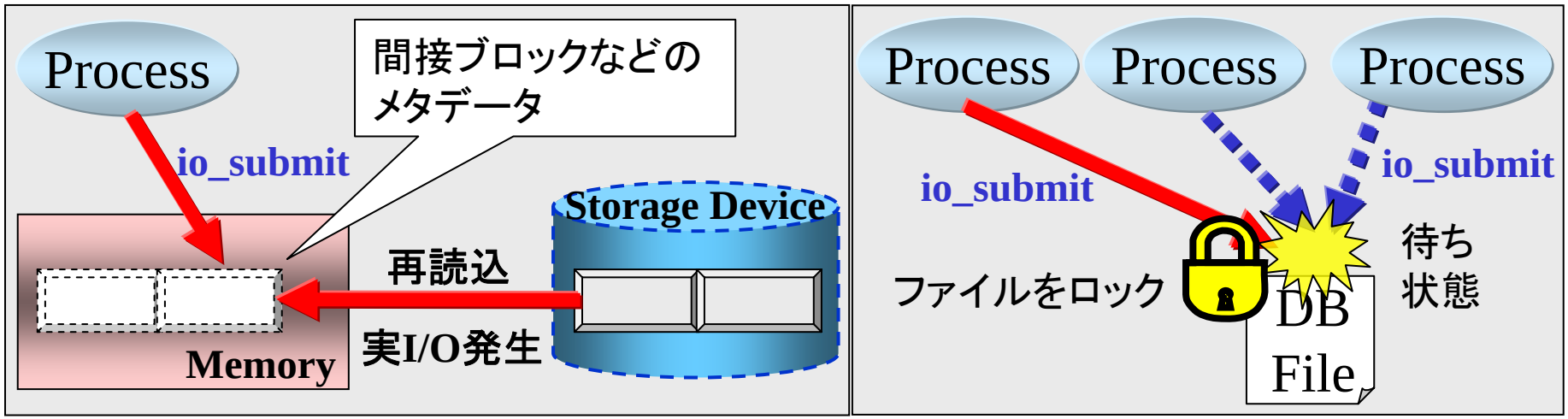
CPUリソースのiowait/idleが発生する原因(2)

■ メタデータが格納されていたページが解放される影響③

□ 非同期I/O処理もメタデータアクセス中、セマフォを獲得してファイルをロック
(Kernel2.4/2.6共通)

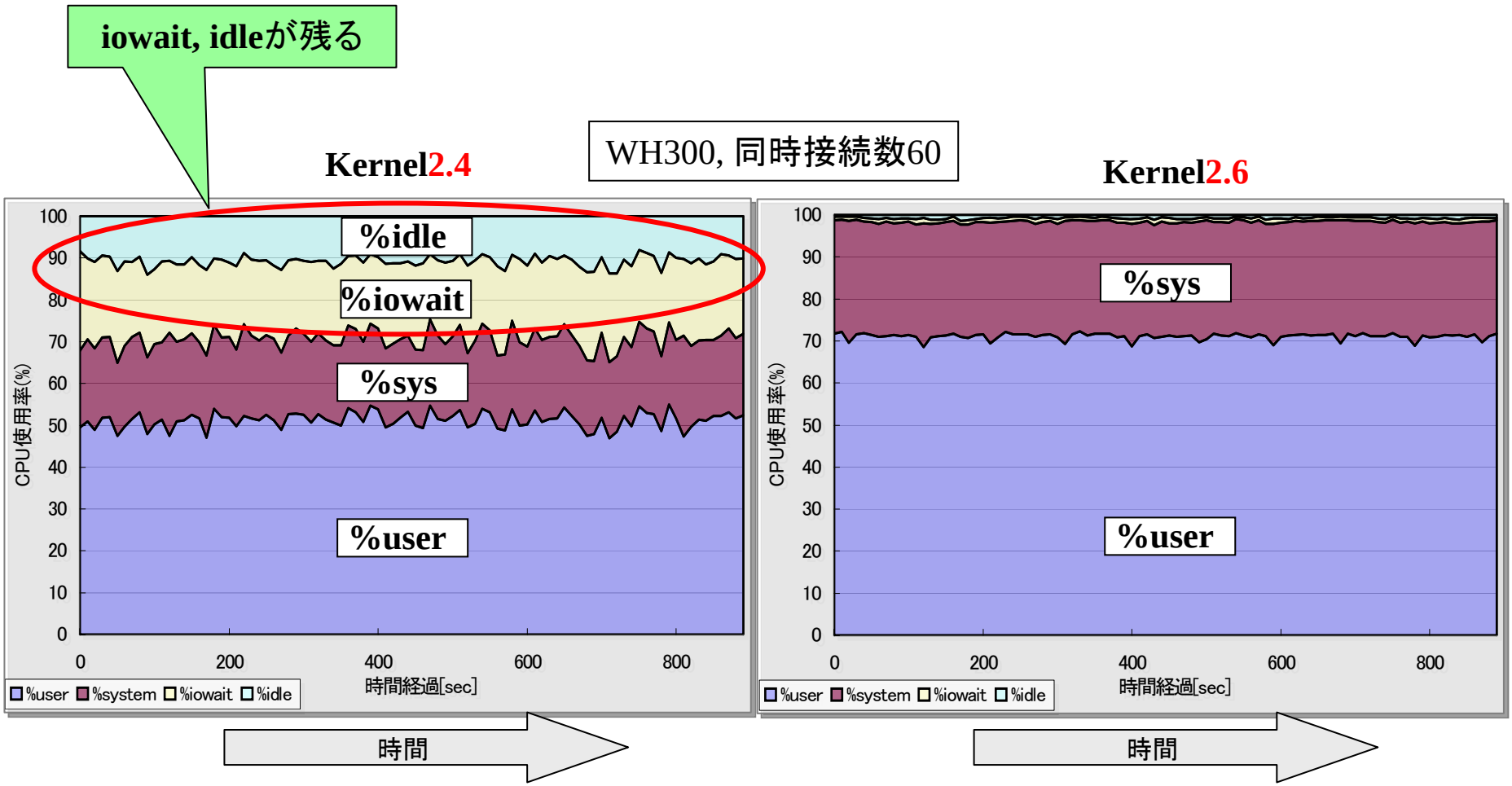
□ メタデータがメモリ上にないため、メタデータの再読込I/O完了待ち合わせでプロセスがブロックされる → CPUリソースの%iowaitの増加

□ 他のプロセスが同一ファイルにI/O要求しても、別のプロセスがメタデータの再読込処理をしている間、ファイルがロックされているため処理が進まない → CPUリソースの%idleの増加



CPUリソースの使用状況

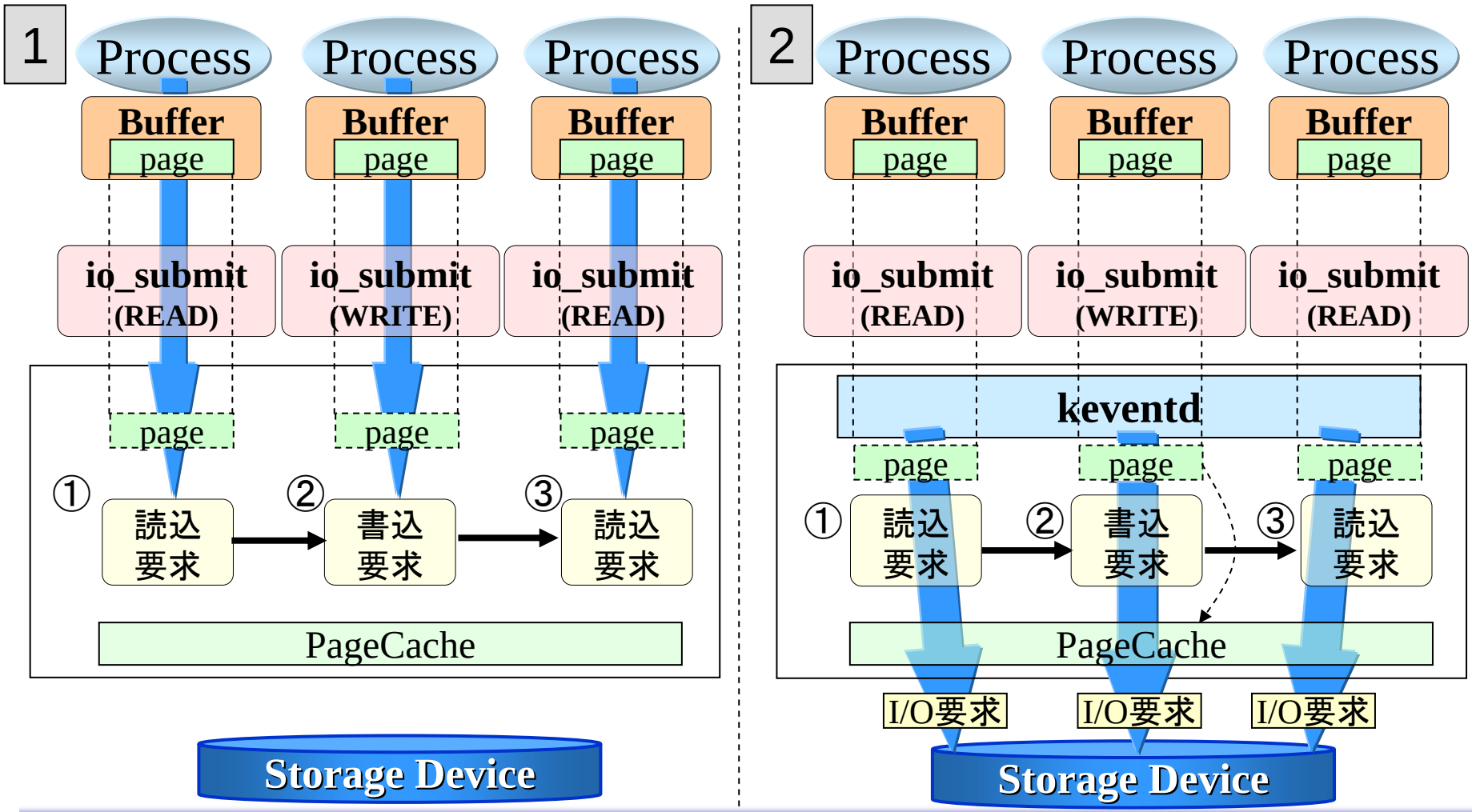
■ CPU使用率の違い(再掲)



ー　参考　ー

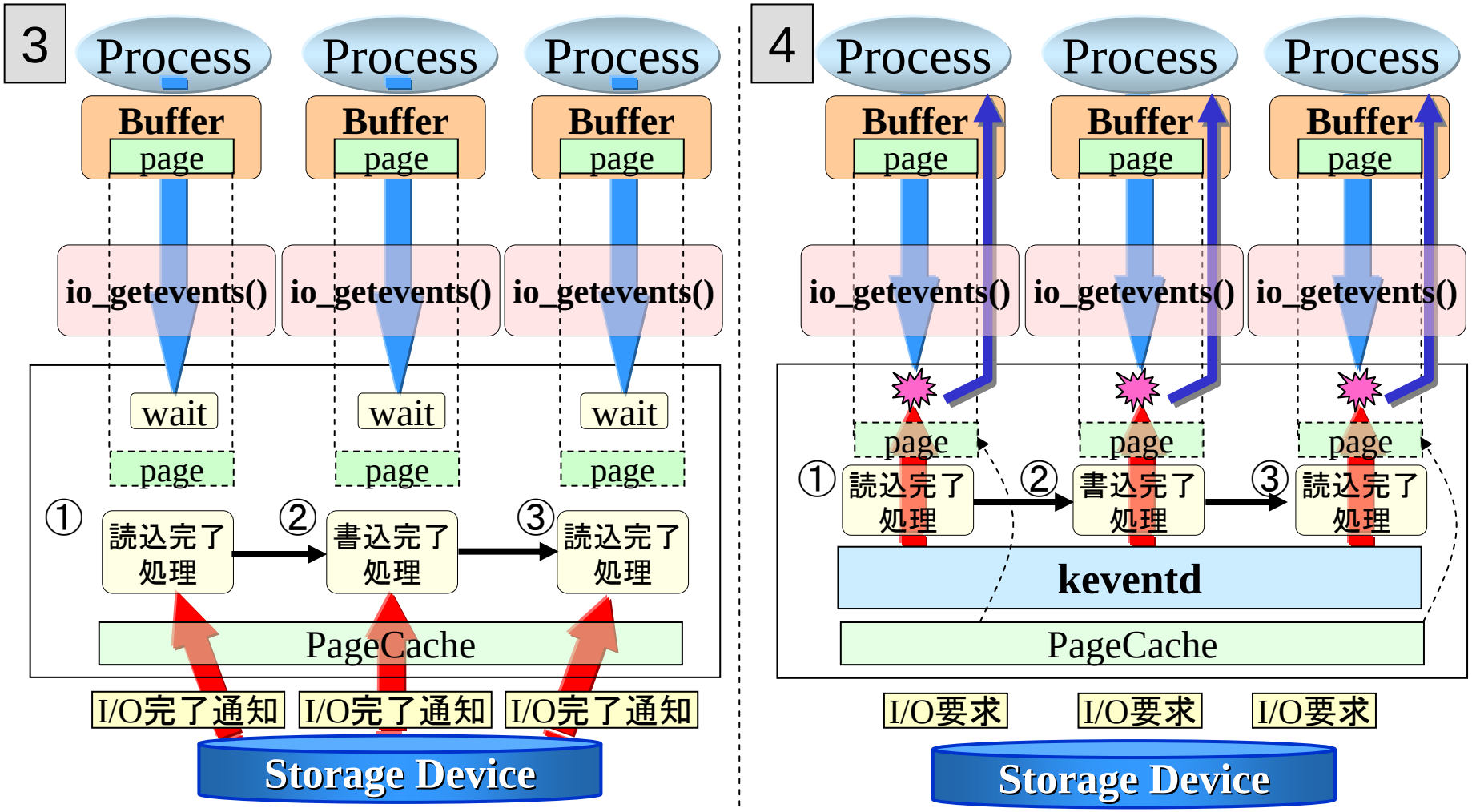
■ Kernel2.4ディストリビューションの非同期I/O(O_SYNC|O_DIRECT)の仕組み

今回のDBMSは主にDBファイル、トランザクションログの書込に非同期I/Oが使用されている



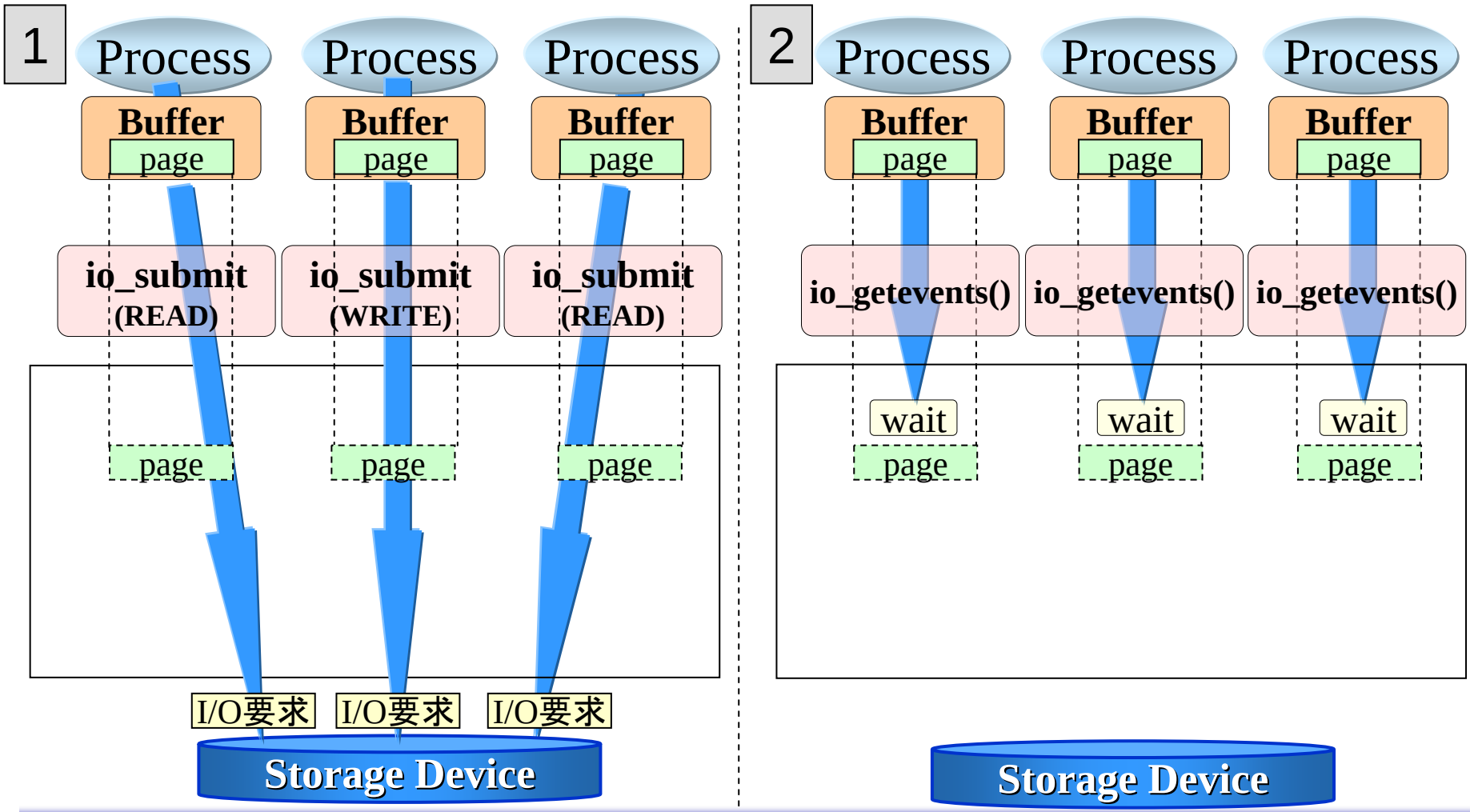
Kernel2.4ディストリビューションの非同期I/OかつDirectI/Oの仕組み(2/2)

- I/O処理を担当するkeventdはシステムに1個のみ → シリアライズされる



■ Kernel2.6ディストリビューションの非同期I/O(O_SYNC|O_DIRECT)の仕組み

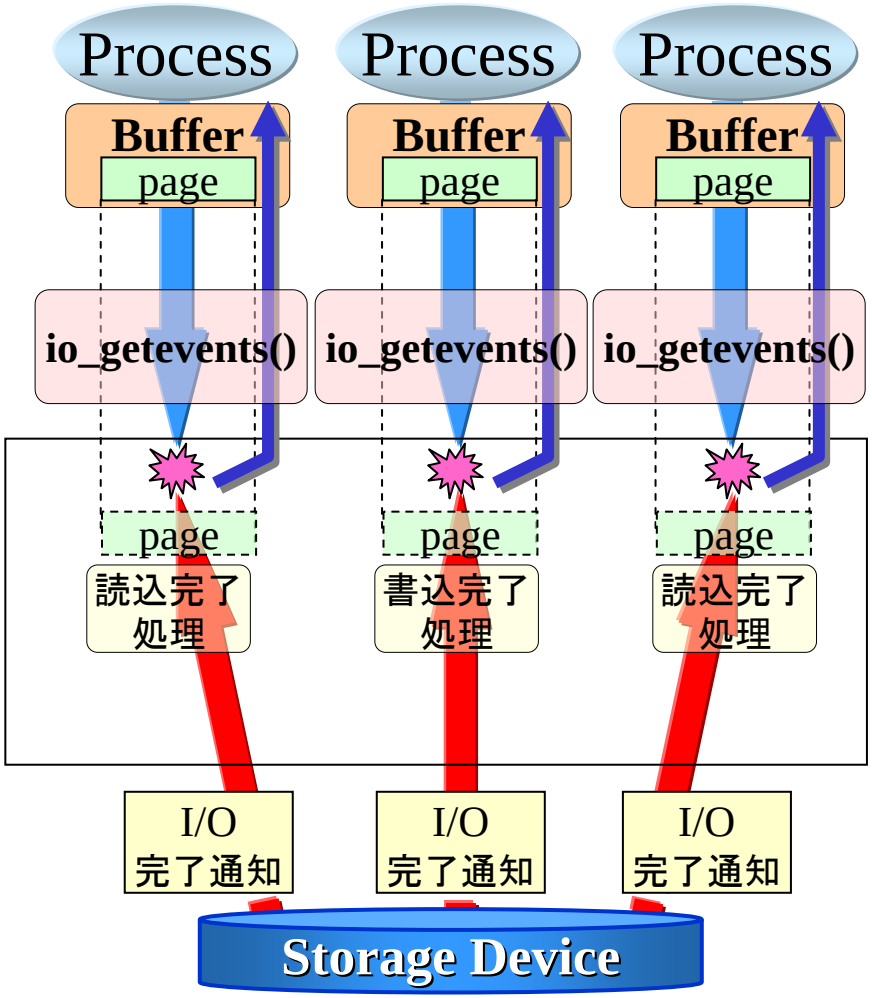
今回のDBMSは主にDBファイル、トランザクションログの書込に非同期I/Oが使用されている



Kernel2.6ディストリビューションの非同期I/OかつDirectI/Oの仕組み(2/2)

□ 各プロセスが個別にI/O処理できる

3



まとめ

■ スケールアップが必要な大規模DBサーバへのLinux適用拡大が期待できる

- Kernel2.6ディストリビューションを使用することで大規模DBサーバにLinuxが適用できる
- Kernel2.4ディストリビューションを使用せざるを得ない場合は、“非同期I/O”の設定は行わず、DirectI/Oのみの設定をする方がよい
  - DirectI/Oが有効になっているか確認すべき
- ページキャッシュを大量に消費する処理は同時に動かさない
  - 非同期I/Oと言えどもメタデータアクセス中はファイルをロックしてしまう
- 運用性を犠牲にしても性能向上を求めたい場合は、rawデバイスを用いることでKernel2.4ディストリビューションの問題点を回避できる
  - rawデバイスの場合、メタデータのアクセスは行われない

謝辞

今回発表した内容の実機検証を実施するにあたっては、
日本電信電話株式会社 NTTサイバースペース研究所
加藤謹詞様、東出治久様に多大なご協力をいただきました。
厚く御礼申し上げます。

ご静聴ありがとうございました。

NTTコムウェア株式会社
オープンソースソフトウェア推進部
野呂 昌哉

e-mail : noro.masaya@nttcom.co.jp
URL : http://www.nttcom.co.jp/

