モバイルゲームインフラあるある物語
～システム障害と技術的対策の例～

スクウェア・エニックス
情報システム部
ソーシャルゲームインフラストラクチャーグループ
野島 貴英

今日は、

「モバイルゲームのインフラのあるある」

物語について語ります！

同業者の皆様におかれましては、

「あるある！」

と共感いただければ。

そうじゃない方々には、

「へぇー！」

と思っていただければ幸いです！

名前：　野島　貴英

仕事：　情報システム部所属。
　　WEB系サーバエンジニア。

実績：　ブラウザゲーム・モバイルゲーム・アーケードゲームのインフラ構築、サーバ技術相談対応・トラブルシュート等。

プライベート:

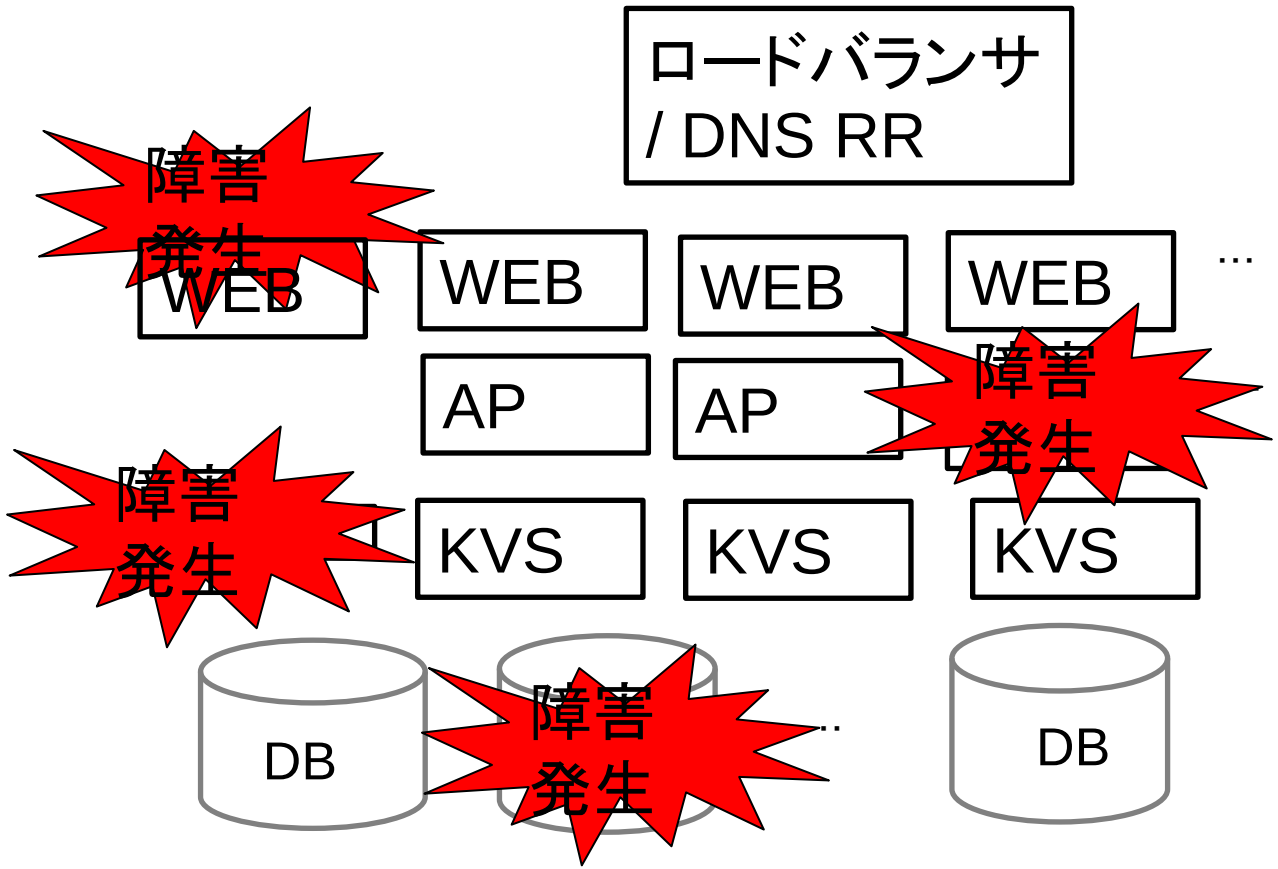
東京エリアDebian勉強会の中の人

http://tokyodebian.alioth.debian.org
(nozzy@debian.or.jp)

-月1回の開発者向け勉強会をやってます
　Debianに興味があれば是非一緒に！
- OpenSourceカンファレンス（東京）でも
　毎回セミナ＆ブースやってます！
　よかったら是非！

ところで本題！

皆さん、おなじみのモバイルサービスインフラの典型例



正直いろいろ発生します！

あるあるその1

負荷分散されているWEBサーバ追加したら、

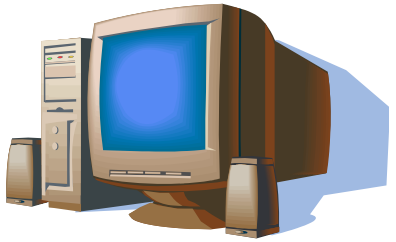
「突然ゲームにつながらなくなった」
（引用：2ch.net/twitterにて多数...）

ヒント：
DNS RRでWEBサーバ負荷分散

DNSのTCPフォールバックが原因でした！

通常のDNS RR の通信

① DNS問い合わせ （53/UDP)
web.foo.bar.com

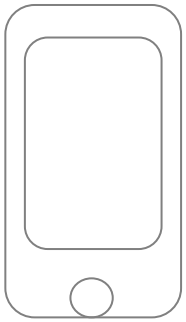


DNS サーバ

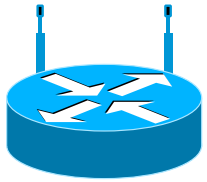
② DNS応答(53/UDP)

123.45.56.78
123.45.56.79
123.45.56.80
....
123.45.56.81

WEBサーバの
IP一覧が返却される

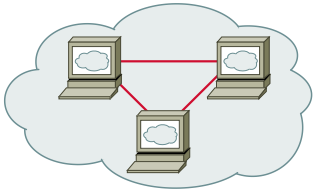


スマートフォン



ブロードバンドルータ

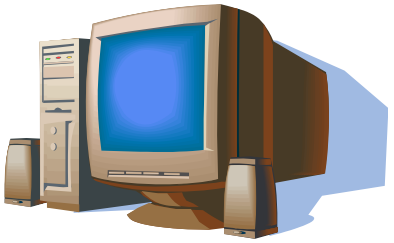
③ DNS応答で得られたIPアドレスのどれかにアクセスしに行く
⇒ブロードバンドルータ毎にまちまちになるので、負荷分散成立！



WEBサーバ群

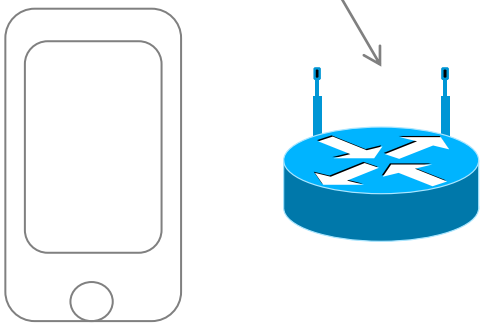
障害発生

DNS サーバー



① DNS問い合わせ （53/UDP)
web.foo.bar.com

ブロードバンドルータ
（結構有名な製品など）

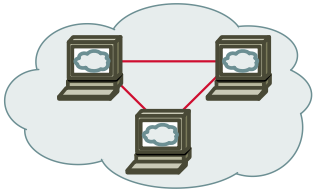


スマートフォン

② DNS応答(53/UDP)
123.45.56.78
123.45.56.79
123.45.56.80
....
123.45.56.81

WEBサーバ追加したら
量が多くて
512バイトを超えた！

③ 512Bytesを超えたのでDNS TCPフォールバックによる問い合わせ(53/TCP)



WEBサーバ群

ブロードバンドルータが対応してなかったり
53/TCPの通信許可していなかったりで通信出来ない

解決策：
ロードバランサーに切り替えました！

思わぬ副作用：
proxyタイプのロードバランサだと、クライアントのIPアドレスが取れなくなったり、ロードバランサ機材の対応帯域を越えてしまったり…

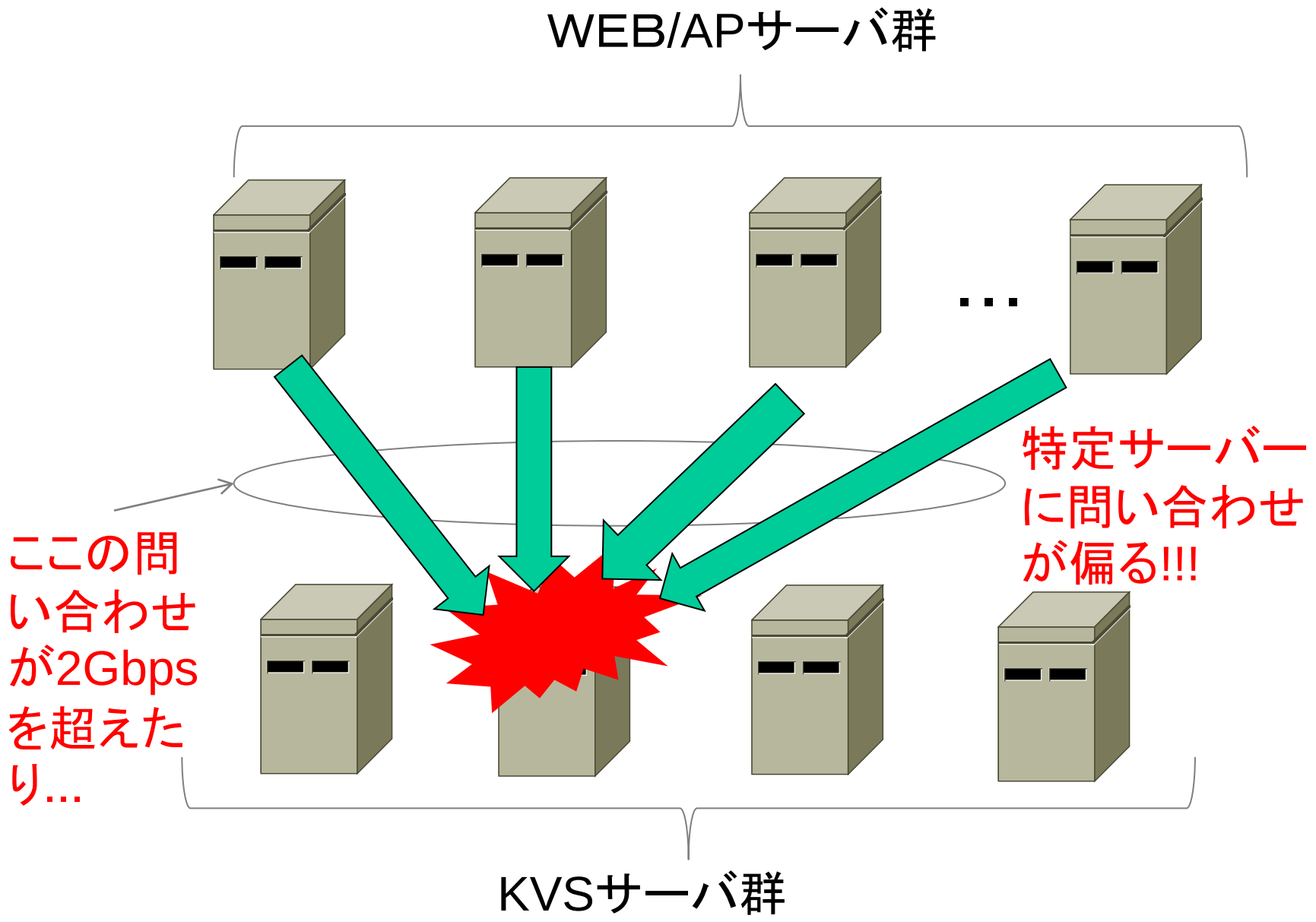
あるあるその2

ユーザ盛況！サーバ増やして対策を...

⇒あれ？頼みのKVSの通信が不安定に？

ヒント： memcached等

アクセスが偏るデータをKVSにいれていたのが原因



技術的解説:

memcachedなどのKVSはキーに紐づいて参照先サーバを固定して使うことが多いです。

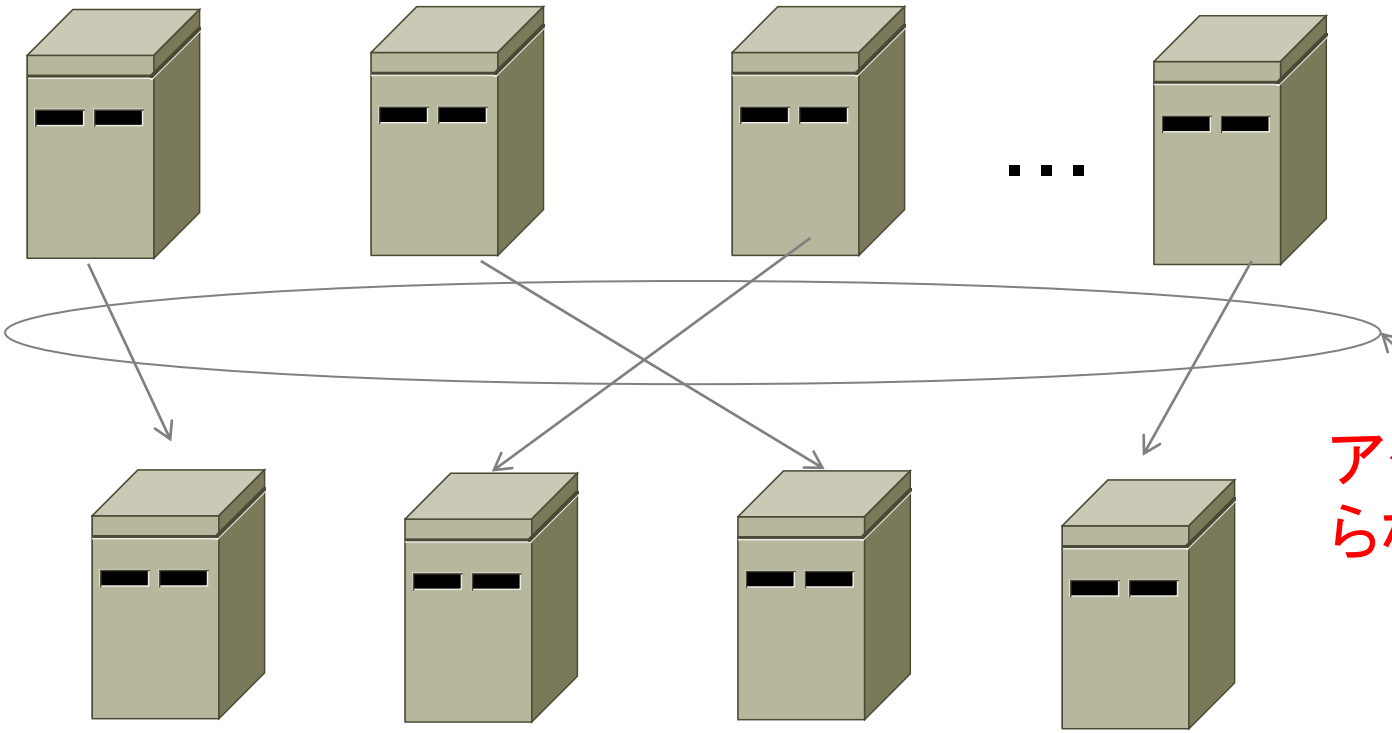
そのため、特定のキーが頻繁に参照されるようなつくりをすると、そのままでは特定のKVSのサーバが極端な過負荷に至る事になります。

解決策その1：

- キーに細工してデータが格納される先を分散

```
// データのセット
for i in 1..10
    set(“common”+i,commondata)
// 参照
get(“common”+int(random(10)+1))
```

WEB/APサーバ群：
プログラムは共通の参照データについて、
キーとしてcommon1～common10をランダムに利用する



common1が配置されている

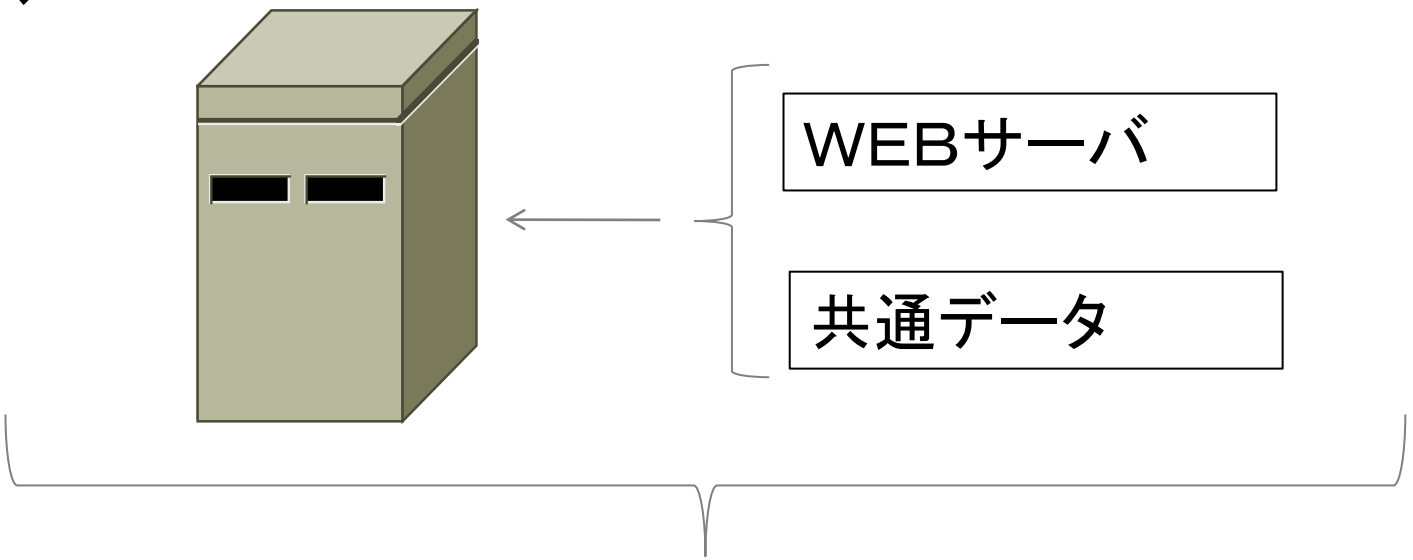
common2が配置されている

common3が配置されている

common4が配置されている

解決策その2:

- WEB/APサーバに共通参照されるようなデータはWEB/APサーバ側に全部置く



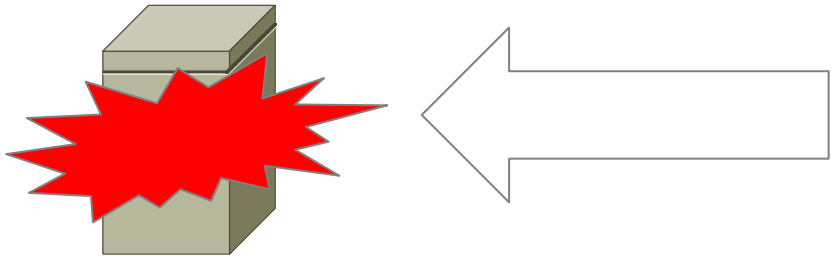
あるあるその3

DBはきれいに水平分割済み!
DBだってスケールするぞ！

⇒ リリースして暫くしたら、負荷もたいした事ないのにサービスが完全に停止。
DBから応答がないんだけど…

分割されたDBの排他制御によるデッドロックが原因でした。

DBの水平分割前

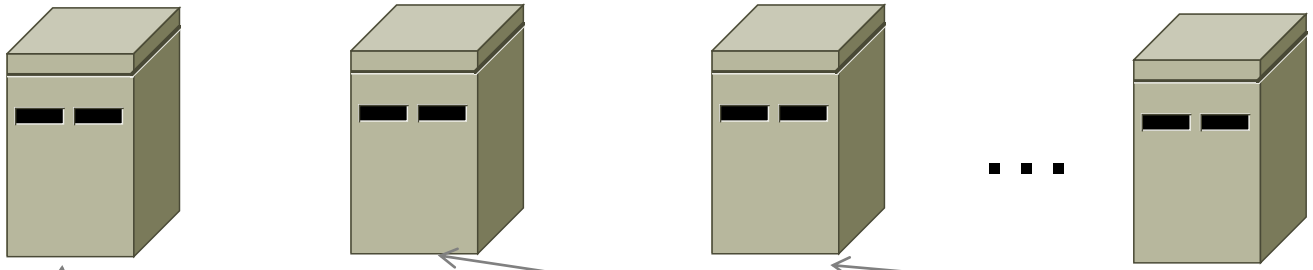


ユーザID	値
1	a
2	aa
…	…
10000	fegefe

DB1個でなんでもやると…

アクセス量が増えすぎると、DBサーバーがあっという間に過負荷に。
とにかく負荷を他にまわせないので、あっという間に対策が打てない。

DBの水平分割後



ユーザID	値
1	a
2	aa
…	…
100	bdcbd

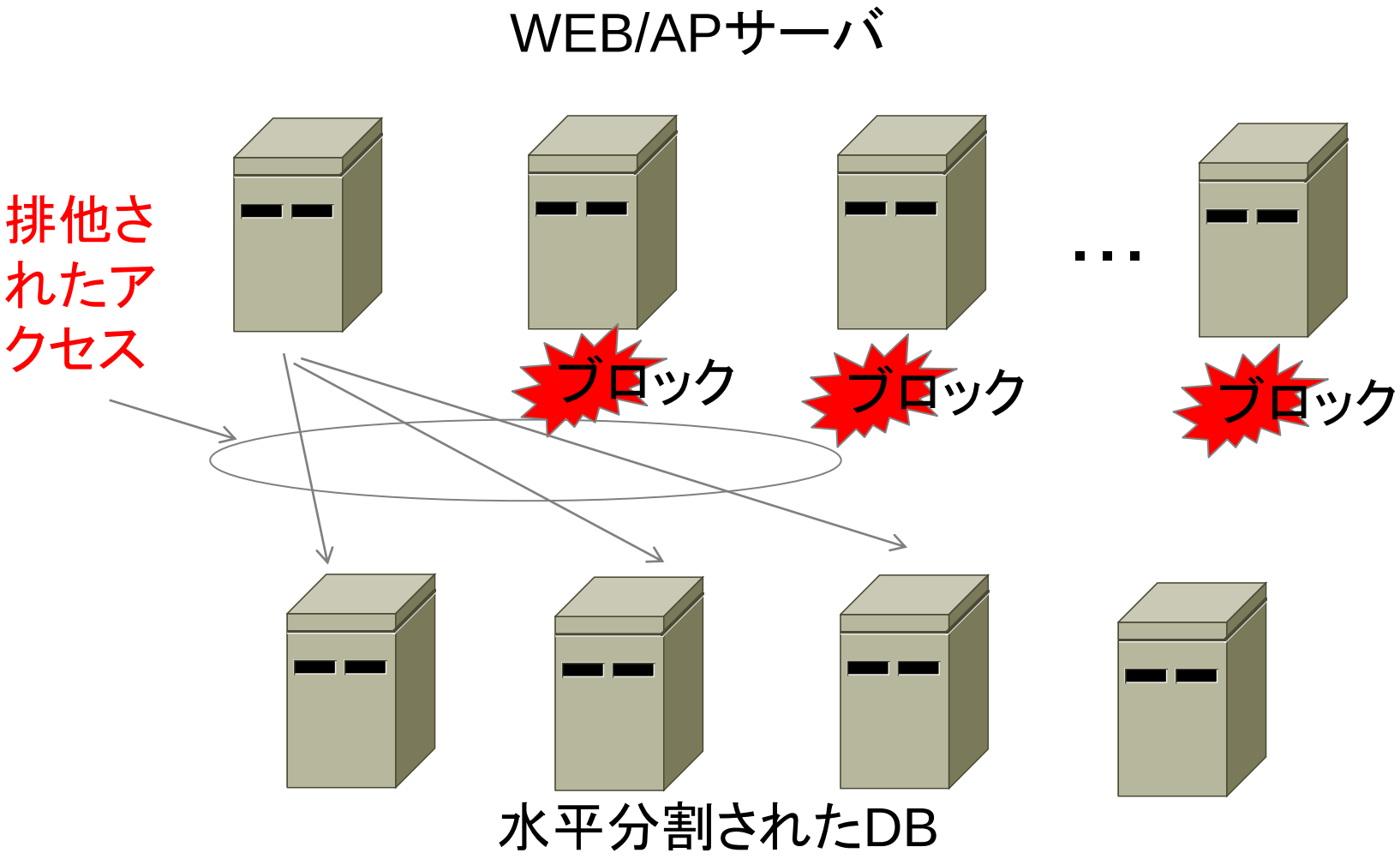
ユーザID	値
101	cdcd
102	babade
…	…
200	fefe

ユーザID	値
301	e3afdfge
302	bfsfsde
…	…
400	fbafaba

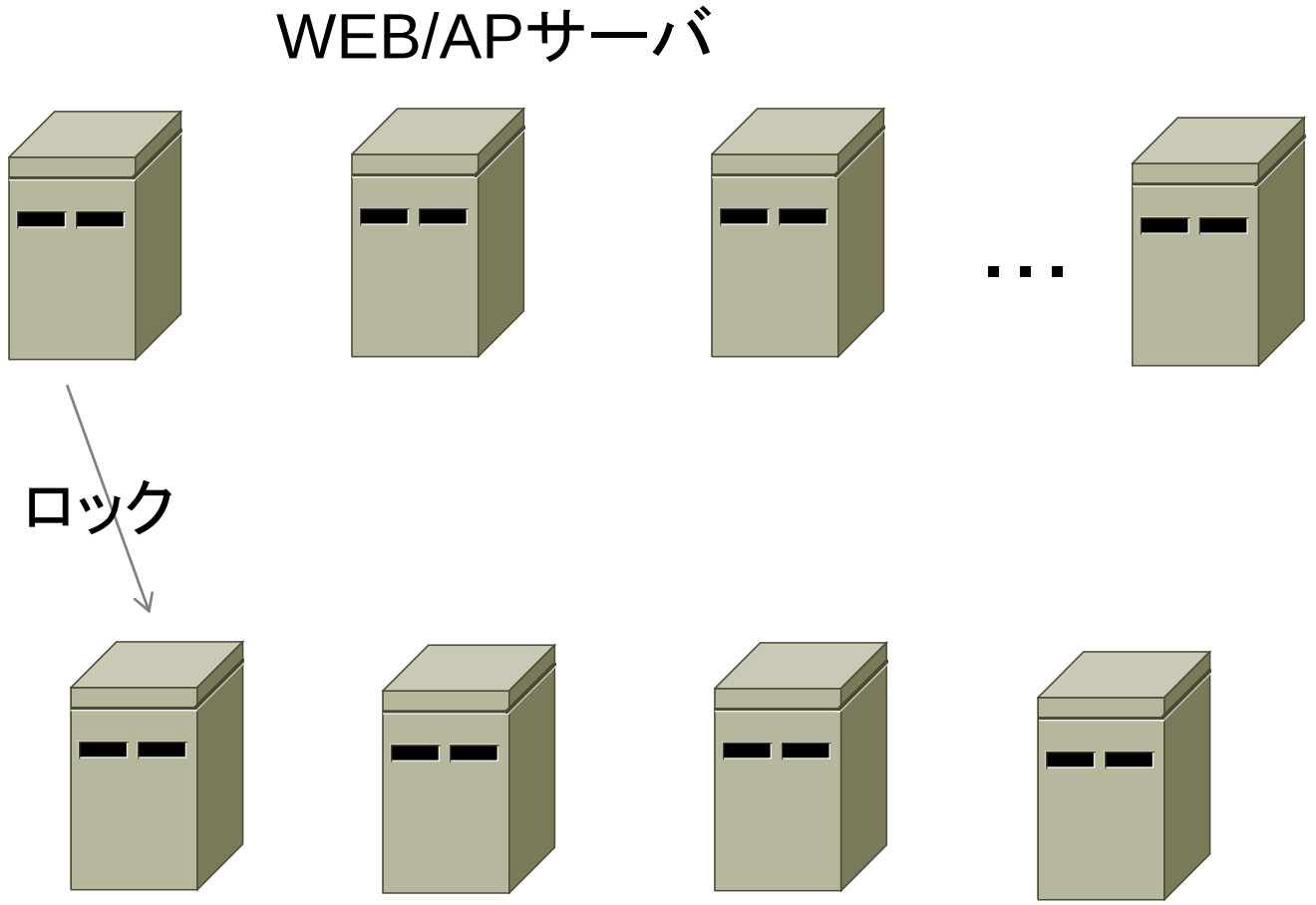
一定のルールでテーブルを分けて、複数のDBに分割して配置
⇒これで処理すれば、DBの負荷を分散できる！

では複数DBにまたがった
データを操作するときは？

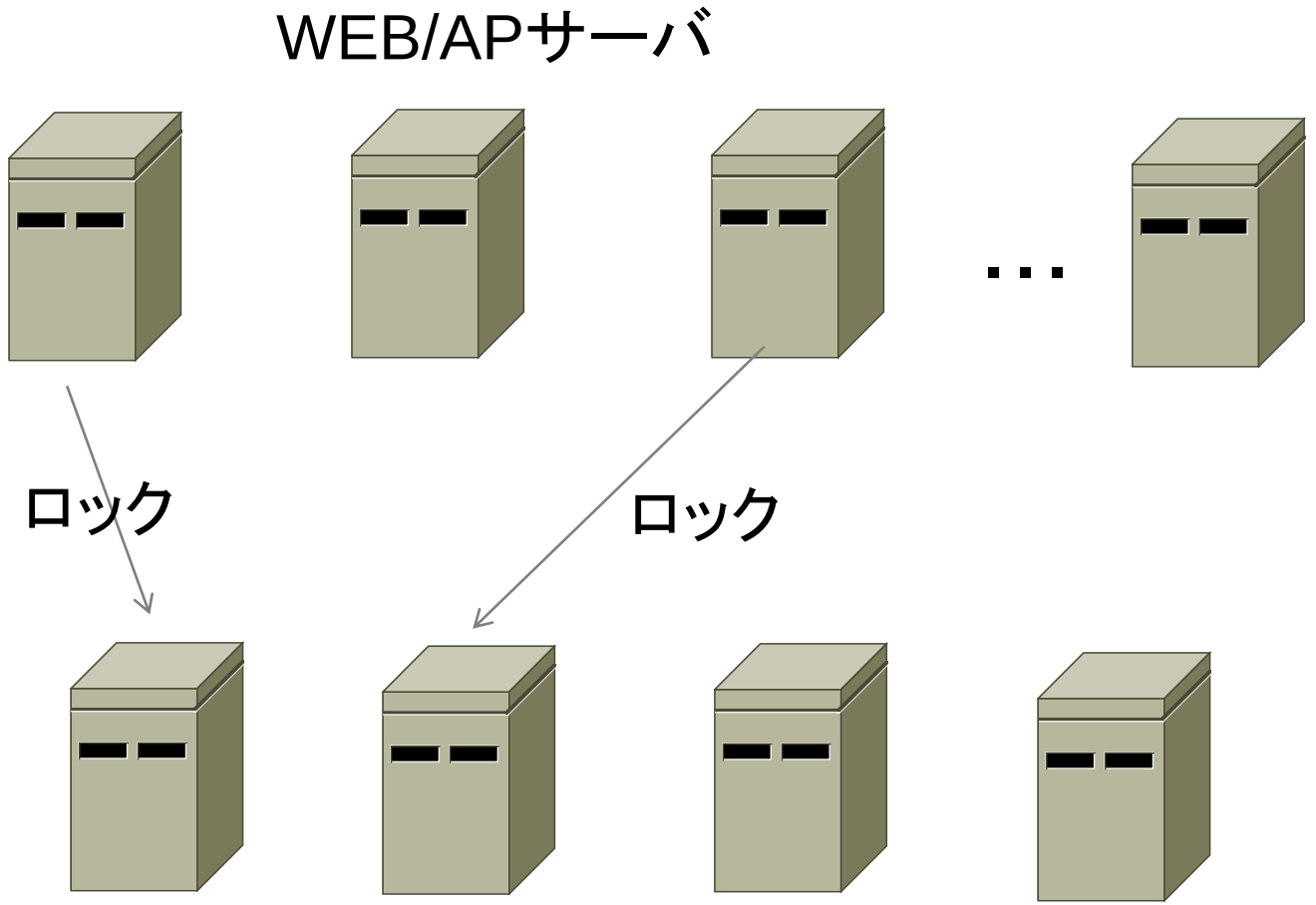
本当はこうなって欲しい



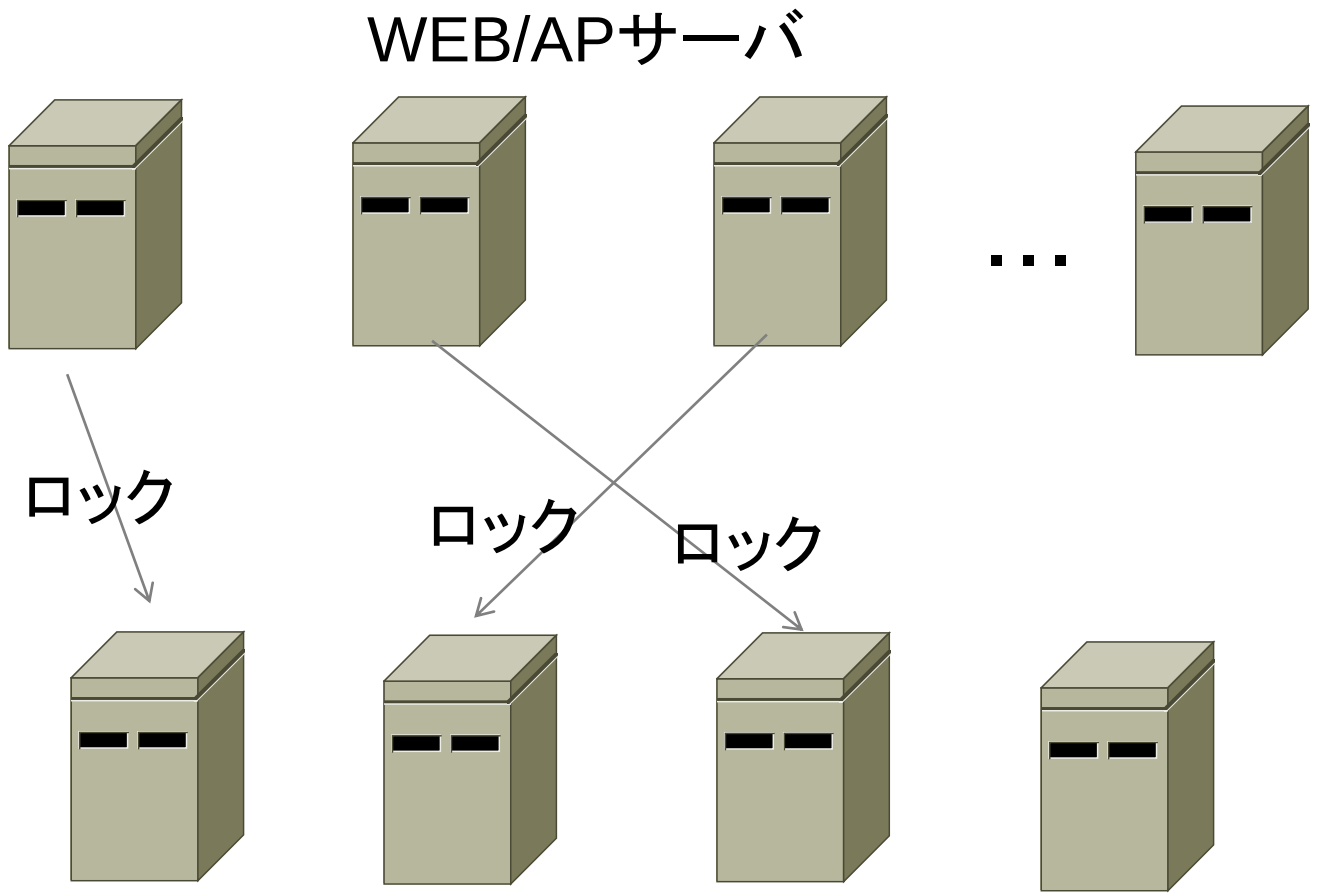
実際に起きた事　Step1.



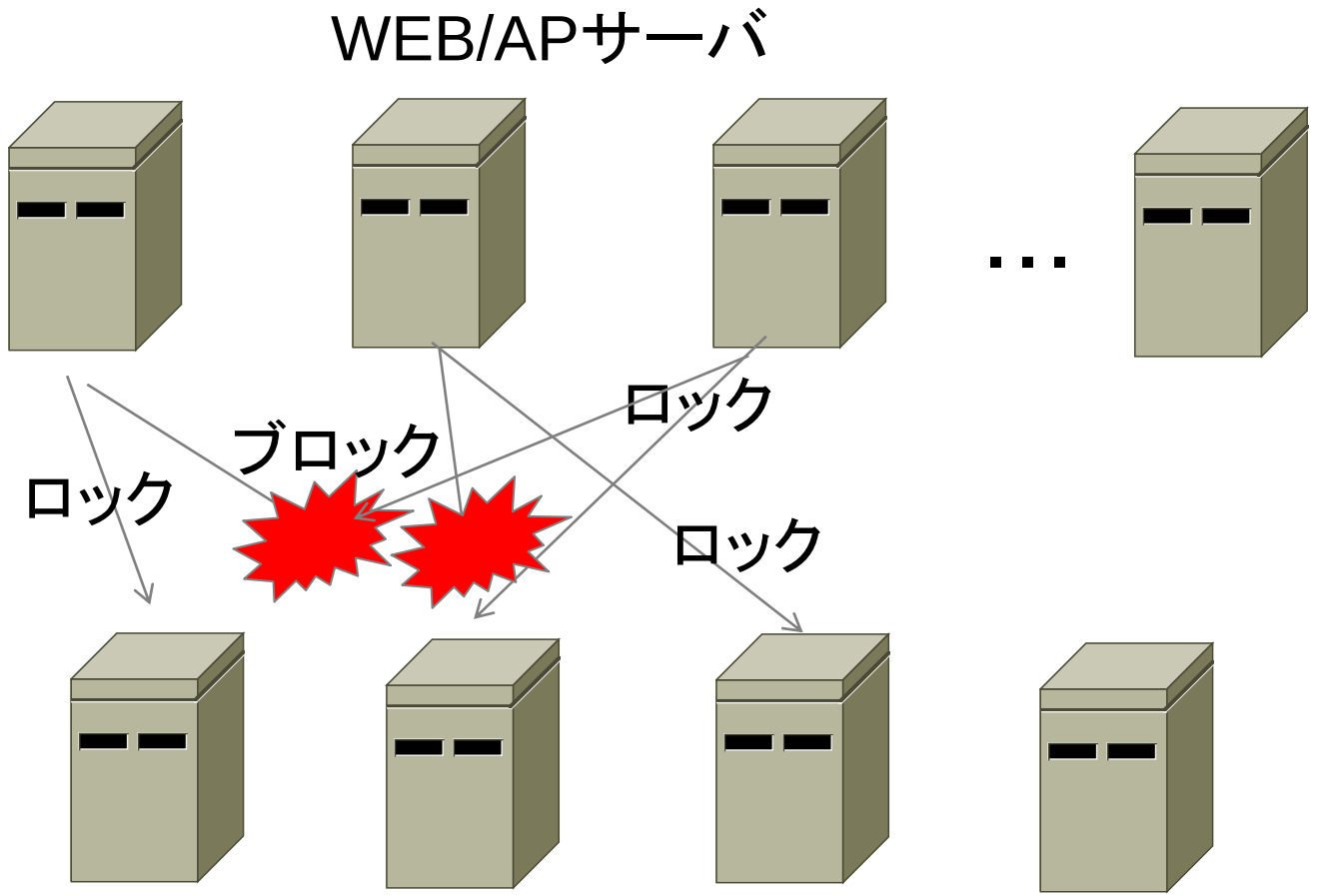
実際に起きた事　Step2.



実際に起きた事　Step3.

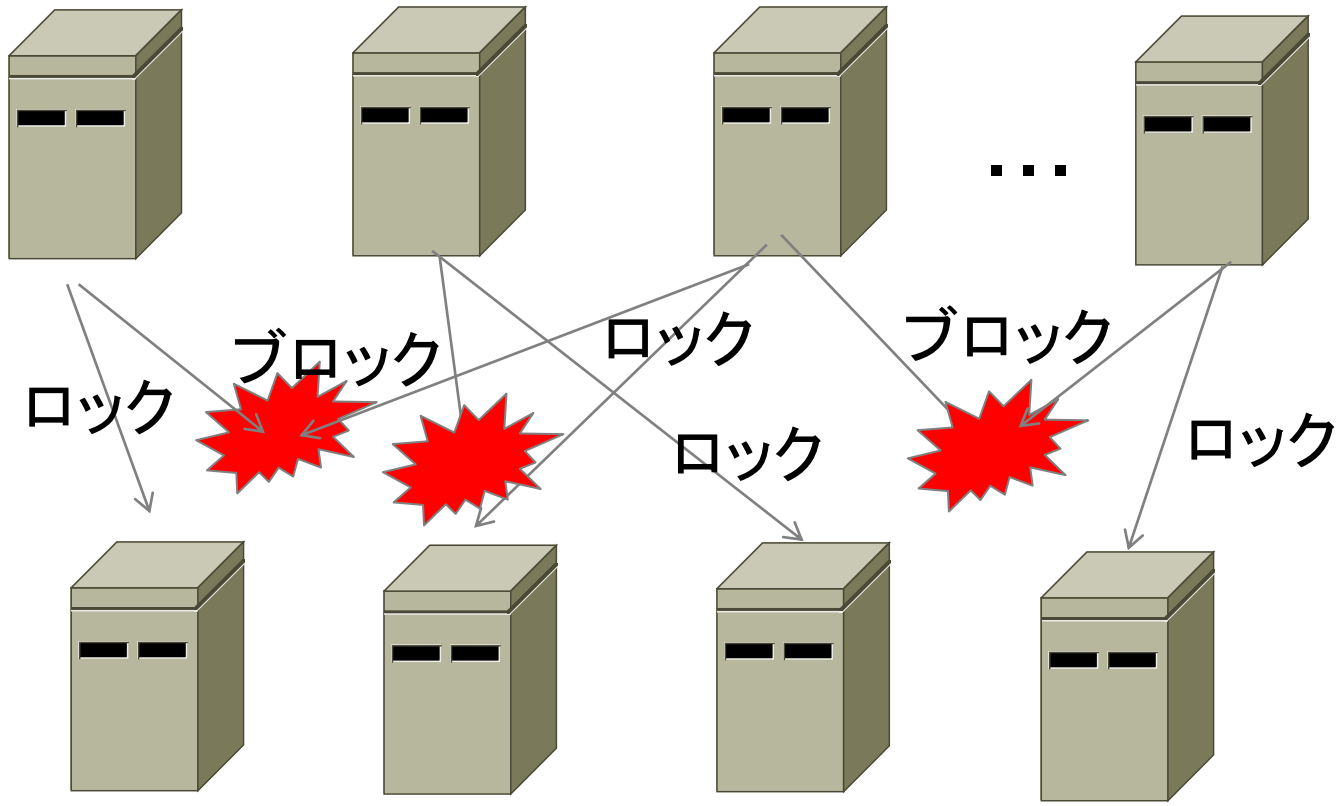


実際に起きた事　Step4.



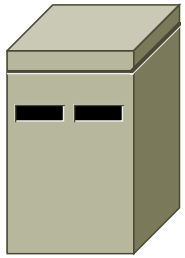
WEB/APサーバがアクセス予定のDBのレコードをロックする前に、他のサーバがDBをロックしてしまう

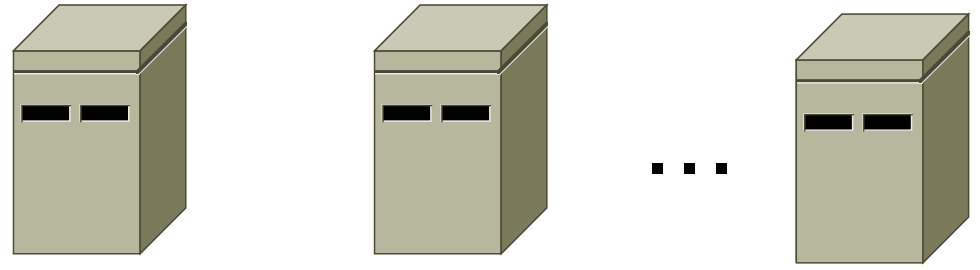
そして誰もデータにアクセスできなくなった



WEB/APサーバがお互いのアクセスを別々にブロック。
最後は誰も結局DBにアクセスできなくなる

解決策：セマフォテーブルを使う





ユーザID	ロック取得サーバー
1	App1
2	App2
…	…
500	None

セマフォテーブル

水平分割されたDB

WEB/APサーバはあらかじめこちらで複数レコードをロック取得。その後、対象の水平分割されたDBへアクセス

現在の課題：

- 水平分割された複数DBのロールバックをエレガントにどう実装するか？
- セマフォテーブルが過負荷にならないのか？

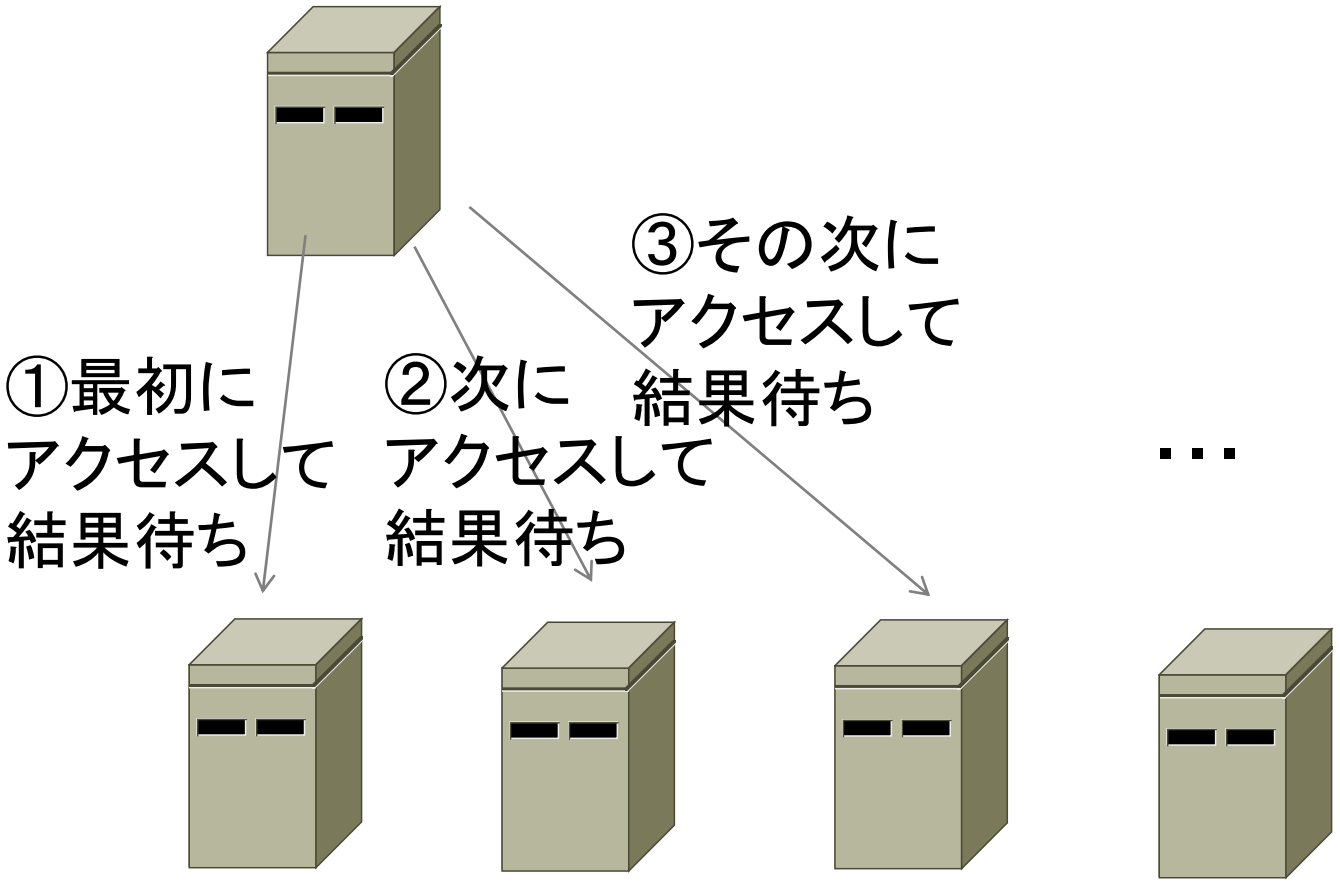
現状：各アプリケーションでがんばって個別設計してます！

あるあるその4

ギルド　vs ギルドを搭載！
DBは水平分割済みだから、DB負荷的にも大丈夫。
⇒
人気のある・人数多いギルドに限ってサービスがままならない。盛り上がるギルドであればある程、さくさくな動きにならない！

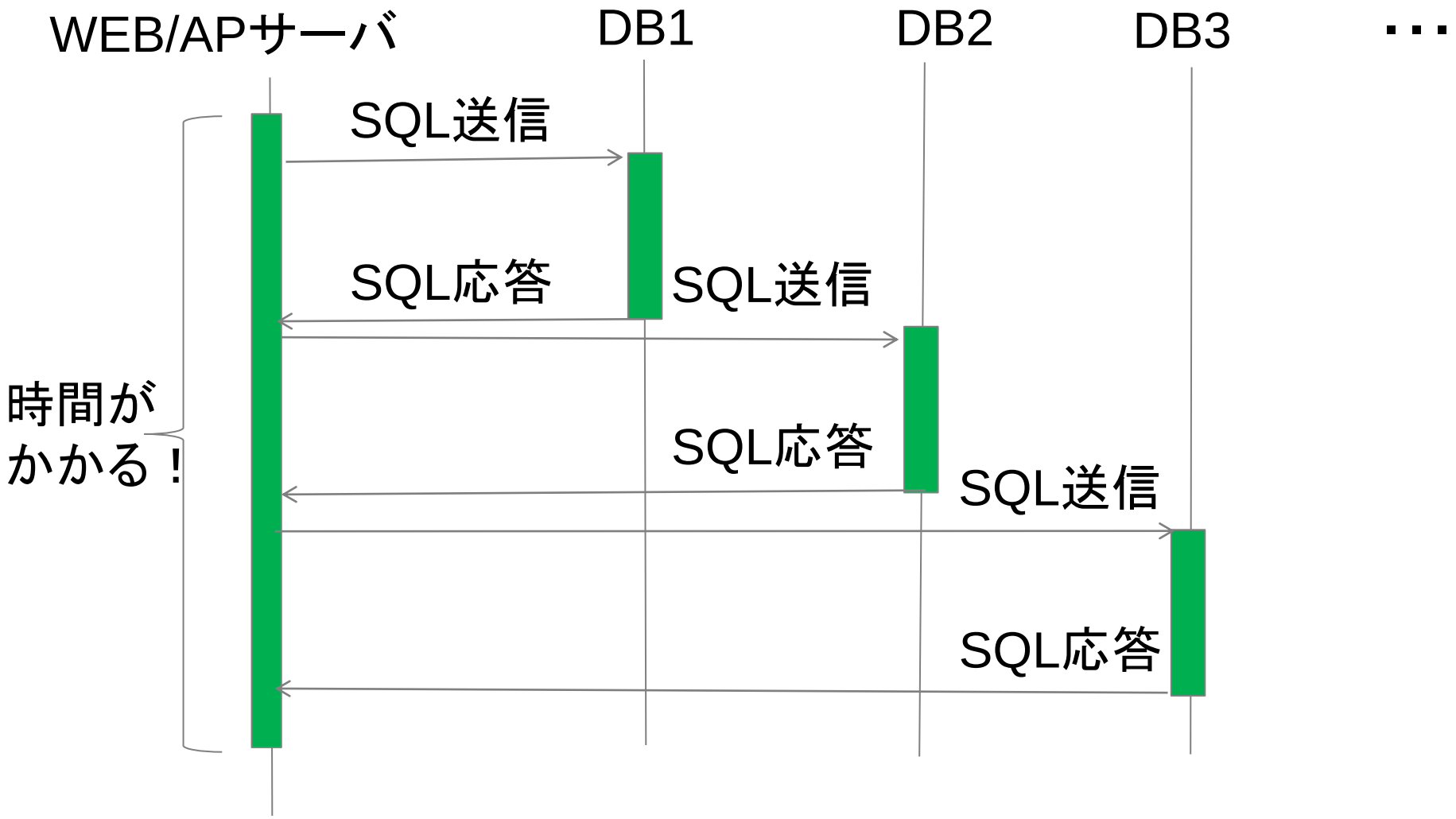
水平分割されたDBの
多数にまたがったアクセスが原因でした

ギルド単位の処理をするのに何も工夫しないと...

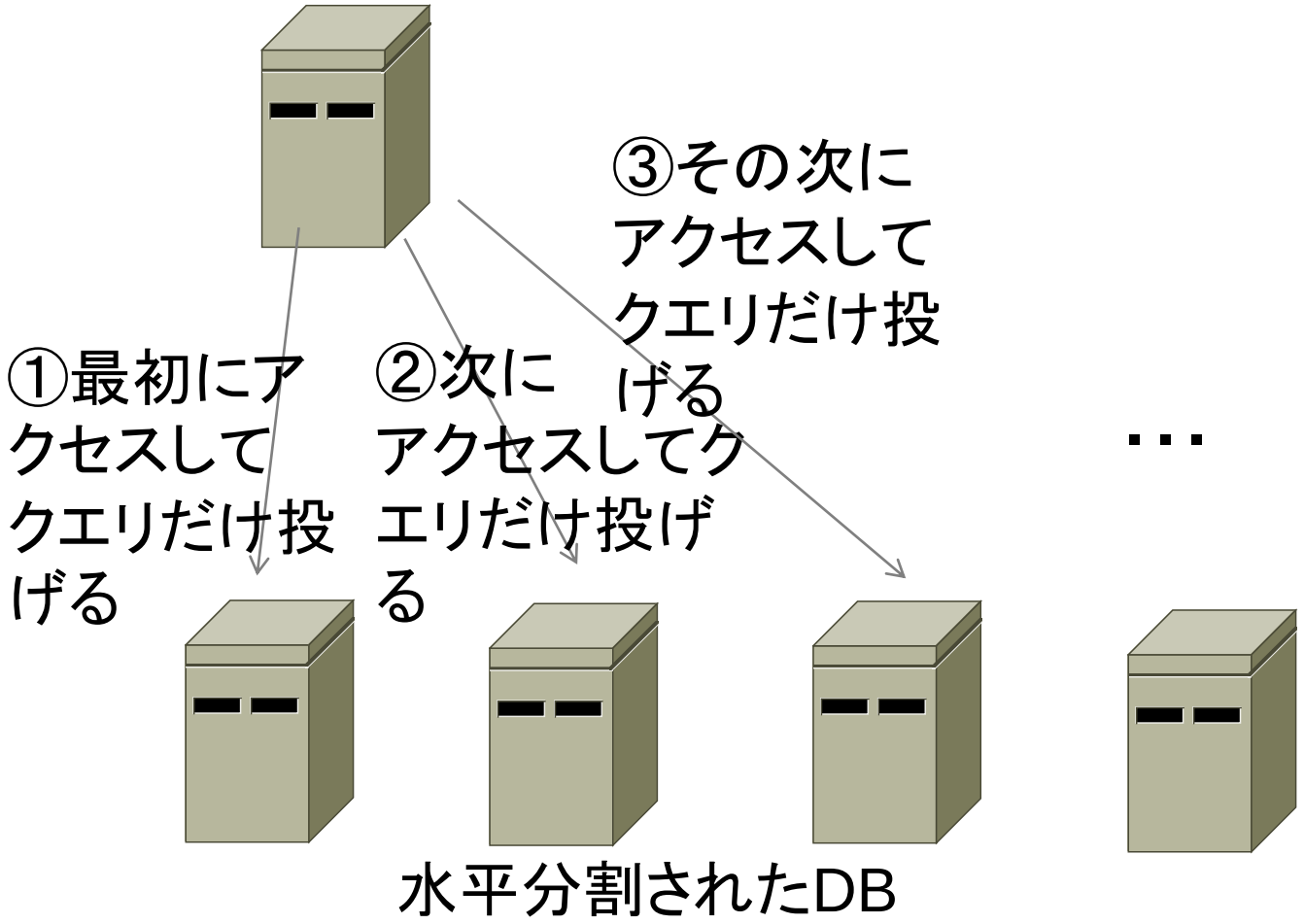


必要なDBデータを集めるのに、DBの台数分時間がかかる、

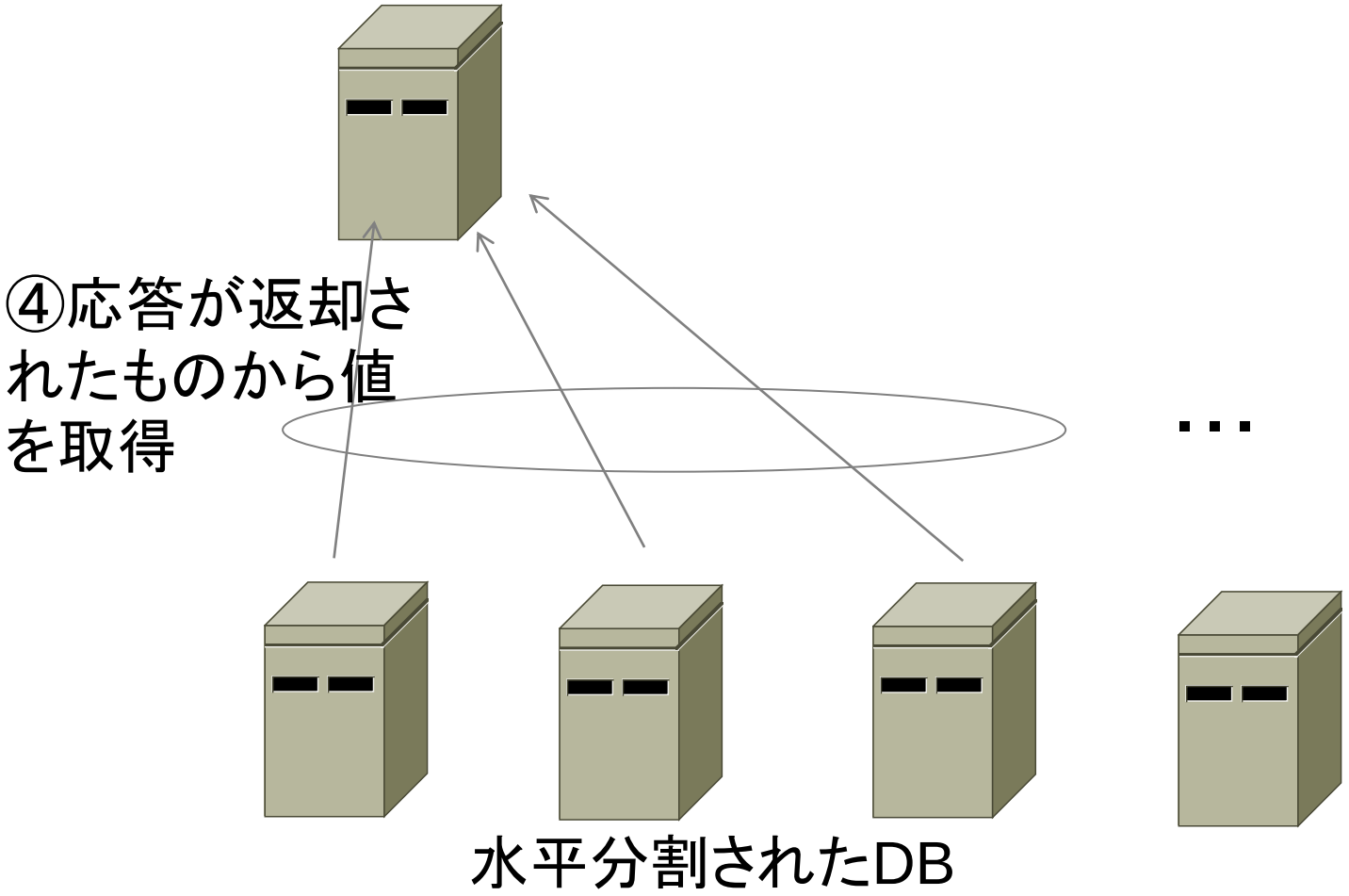
あるあるその4



解決策その1： 並列でDBクエリを投げる

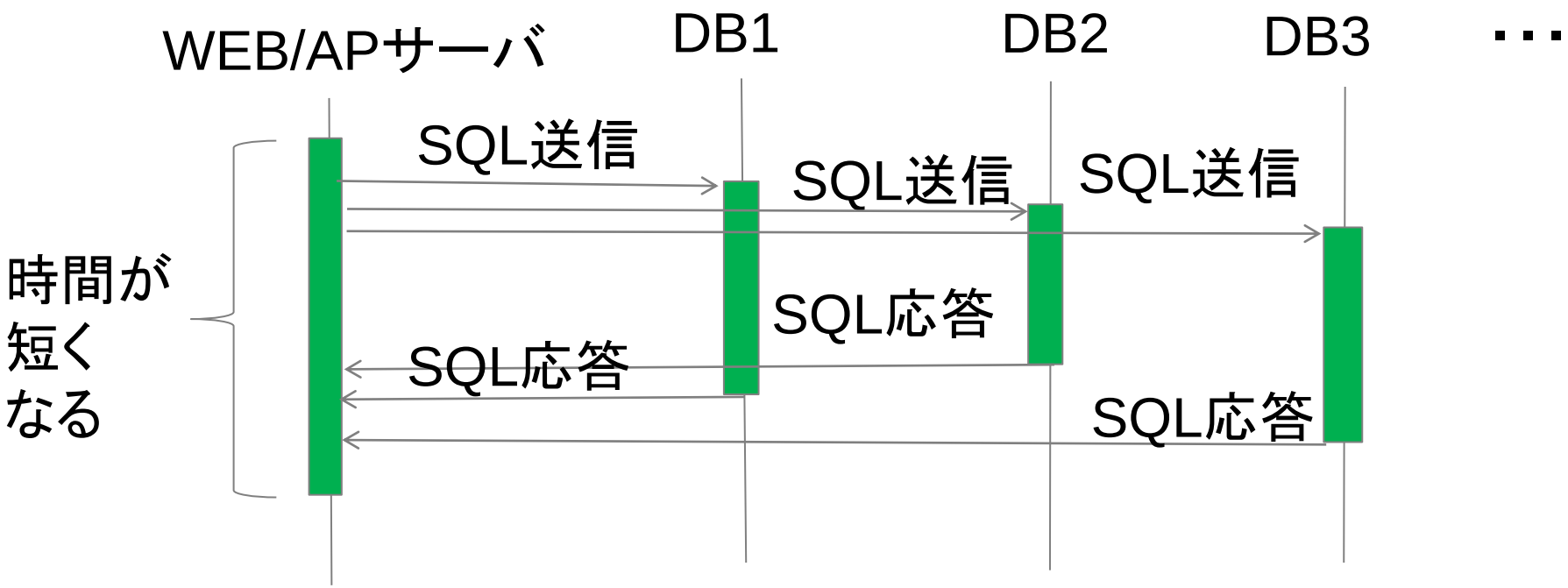


解決策その1： 並列でDBクエリを投げる（続き）



必要なDBデータを集めるのが効率的になる。

解決策その1： 並列でDBクエリを投げる（続き）

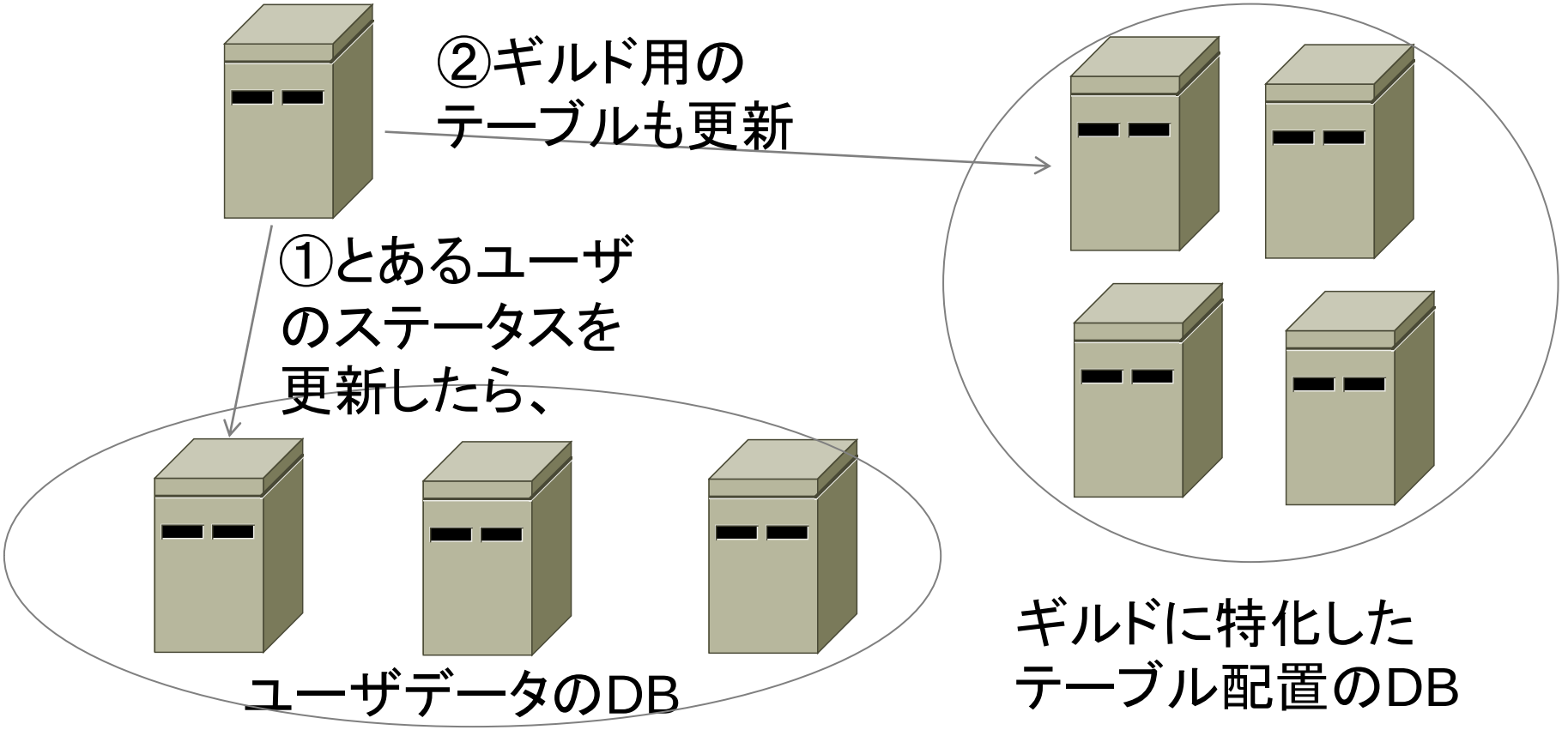


解決策その1について補足：

サーバ側の開発言語では並行してSQLを送信するのが苦手なOSSプロダクト/言語があります。
例：PHPなど

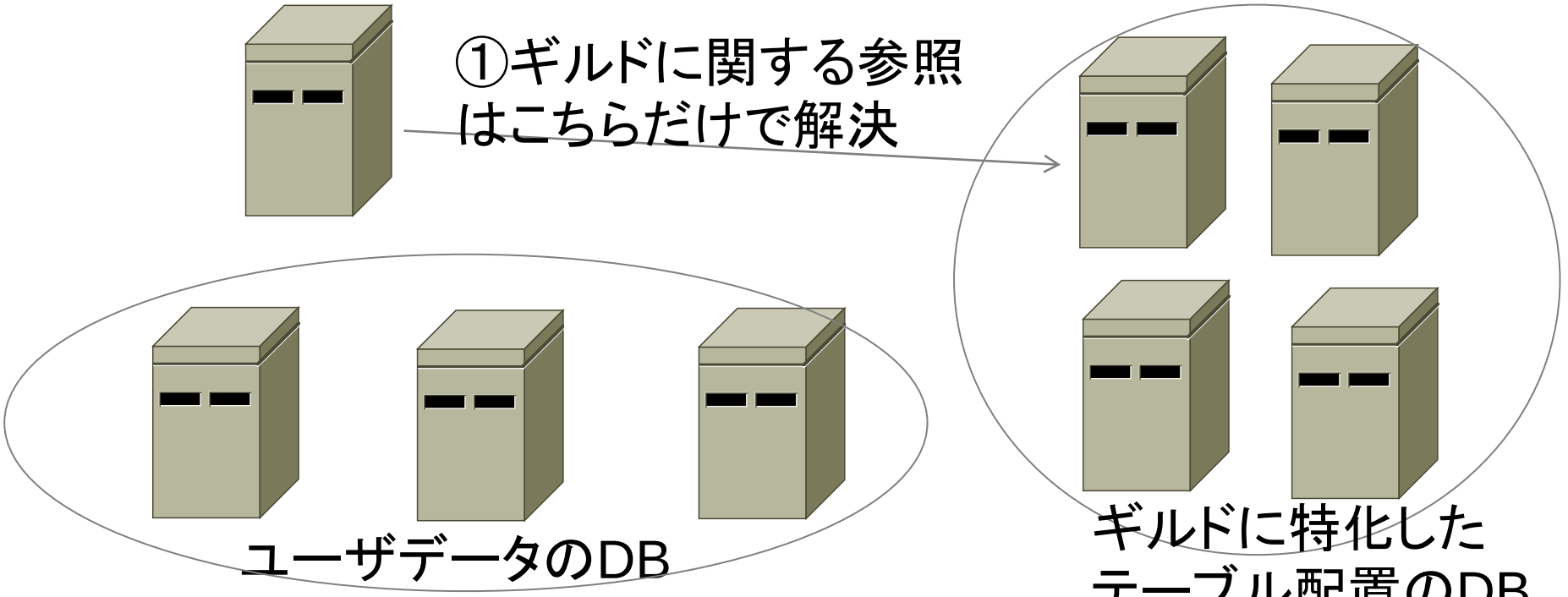
思い切って、並行してSQLを送信する言語で処理するなどを検討する必要があります。

解決策その2：ギルド処理を別のDBで持つ



ギルド処理であっても更新は少ないことが多い。

解決策その2：ギルド処理を別のDBで持つ（続き）



ギルド毎にデータを参照であれば参照すべきDBを減らせる。特に
参照量＞＞更新量
の場合が多いのでパフォーマンスは非常に良くなるケースが多い

解決策その2について補足:

　サーバ側の開発言語では並行してSQLを送信するのが苦手なOSSプロダクト/言語を使わざるを得ない場合に有効です。

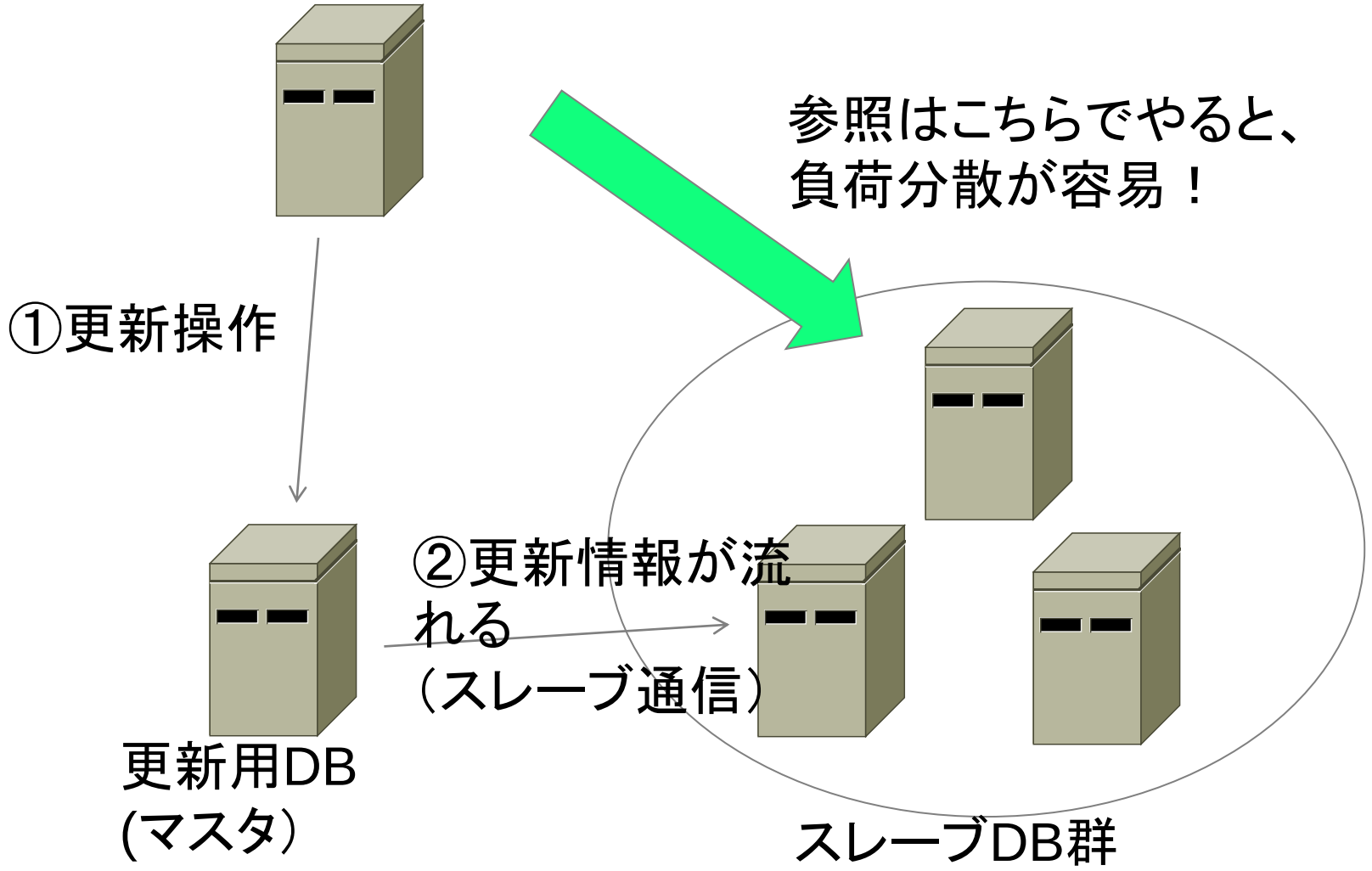
　内容次第ではギルドプレイの仕様を調整する必要がでてくる場合があります。

あるあるその5

DBのマスタスレーブ構成にし、参照系は全部スレーブに流しているから負荷対策はばっちり！

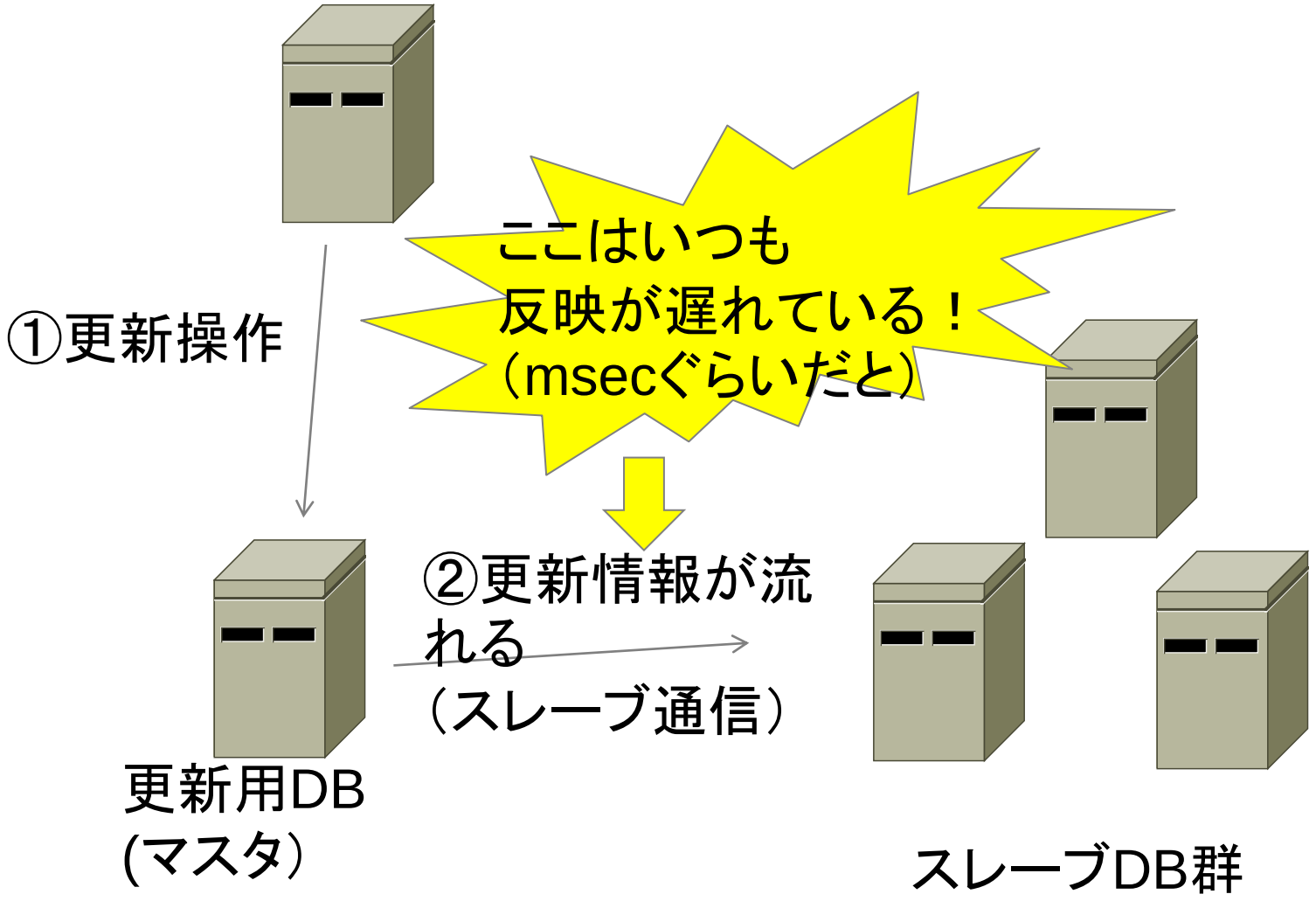
⇒負荷が高くなると、マスタのデータが不安定に壊れていく...

DBのマスタ・スレーブ構成

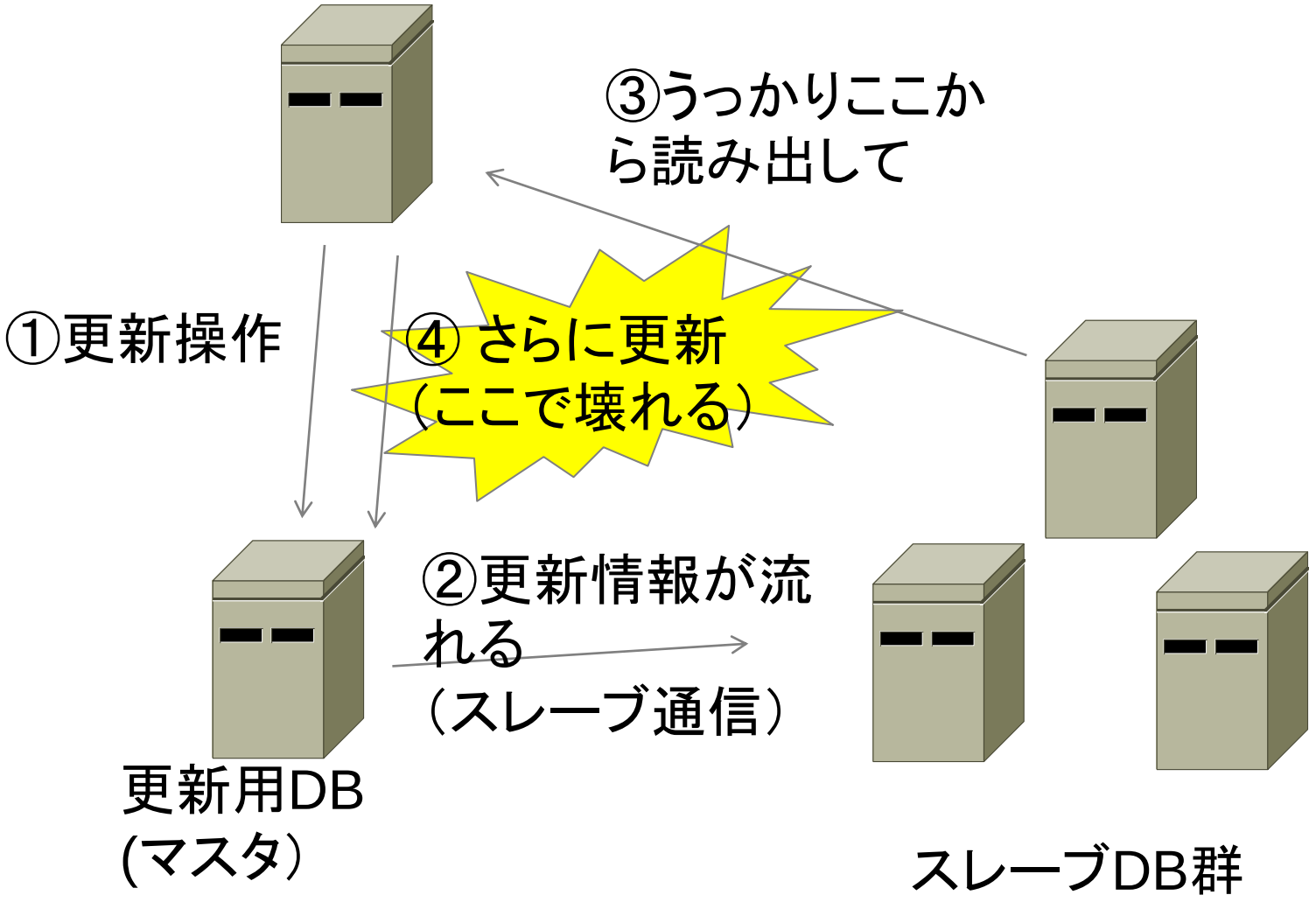


遅延が全く許されないデータのリードまでスレーブでやっていたのが原因でした！

DBのマスタ・スレーブで起きることの実際



DBのマスタ・スレーブで起きることの実際（つづき）



技術的解説:

OSSプロダクトのDBの場合、スレーブDBから読み出した結果を元にマスタを更新すると高負荷時に限って壊れます。しかも、うっかりこの構成が出来上がってしまい、リリース時に判明ということも。

対策:

遅延が全く許されないデータのリードはマスタ側で行なうようにしましょう。

その他あるある

リアルタイムの対戦を搭載。あるいは、
MOが出来た！リリースしたら大人気！

⇒結果、
月のネットワーク課金がすごい事に…

原因：通信パケットの罠

解説：

リアルタイムでデータ交換すると、小さいパケットが大量に飛び交うことになります。I

パブリッククラウドだと、通信量で課金が殆どであり、所謂CDN/キャッシュも効かないので、全部通信費用に跳ね返ります。通信費用に注意がいります。

サーバとしては全部冗長済み。
サーバ1台ぐらい停止しても大丈夫！
　⇒
サーバ1台リブートしたら、ありえないほどのサーバ負荷が！

…サービスが長時間とまっちゃった…

原因:
セッション切れ再コネクトラッシュ事故の恐怖

解説:
アクセス量が非常に多いサービスで、KVSなどのサーバがリブートして、セッションが大規模に無効化。結果、ユーザは一斉に再ログインをする事に。

全ユーザのログインラッシュは通常想定しないので、一瞬でシステムダウン。その後、ユーザの再ログイン、システムダウン....の無限コンボが。

負荷テスト中に障害テストの実施をお勧めします。

モバイルゲームサーバで「あるある」な話を主に語ってみました。

　もっといろいろありますが、大規模モバイルゲームのインフラエンジニアのあるある話を大募集中！