社内勉強会

NGINX誕生の歴史

目次

- NGINXとは
- ApacheとC10K問題☆
- NGINXの誕生
- まとめ

NGINXとは

NGINX

プロフィール

・WEBサーバー

・2004年生まれ

得意な事

・大量の同時アクセス耐える事 など

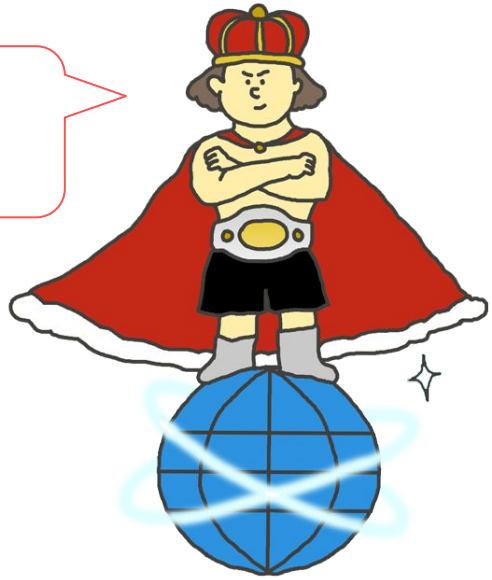
苦手な事

・動的コンテンツの処理(スクリプト言語の処理) など

Apacheと C10K問題

APACHE
HTTP SERVER PROJECT

圧倒的シェア！



プロフィール

・WEBサーバー

・1995年生まれ

得意な事

・動的コンテンツの処理(スクリプト言語の処理) など

苦手な事

・大量の同時アクセス

Apacheは1990年代後半、
圧倒的なシェアを誇っていたが
その後インターネットが普及し...

C10K問題

=クライアント 10000台 問題

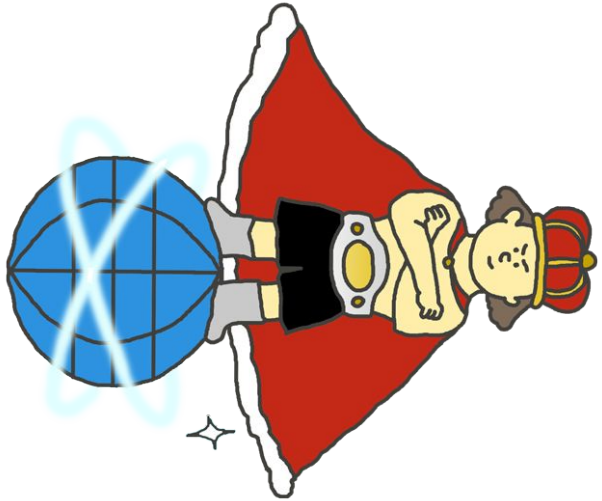
原典引用:*So hardware is no longer the bottleneck.*

➔クライアントが増えた時に
限界でない理由でのパフォーマンス劣化

ハードウェアの

FD,PIDの枯渇、同時起動可能なプロセス数の上限**に達し 前の処理が終るまでプロセスを起動出来なくなる**

Apache
(当時は)C10K問題に対抗できなかった...



なぜApacheはC10K問題に対応できなかったのか

原因はApacheのリクエスト処理方法の特徴にあった

MPM(Multi Processing Module)

プロセスを複数生成することで同時にリクエストを処理出来る。

これを実現するモジュール（追加機能）の事

➔同時リクエストが来た際の処理の方法

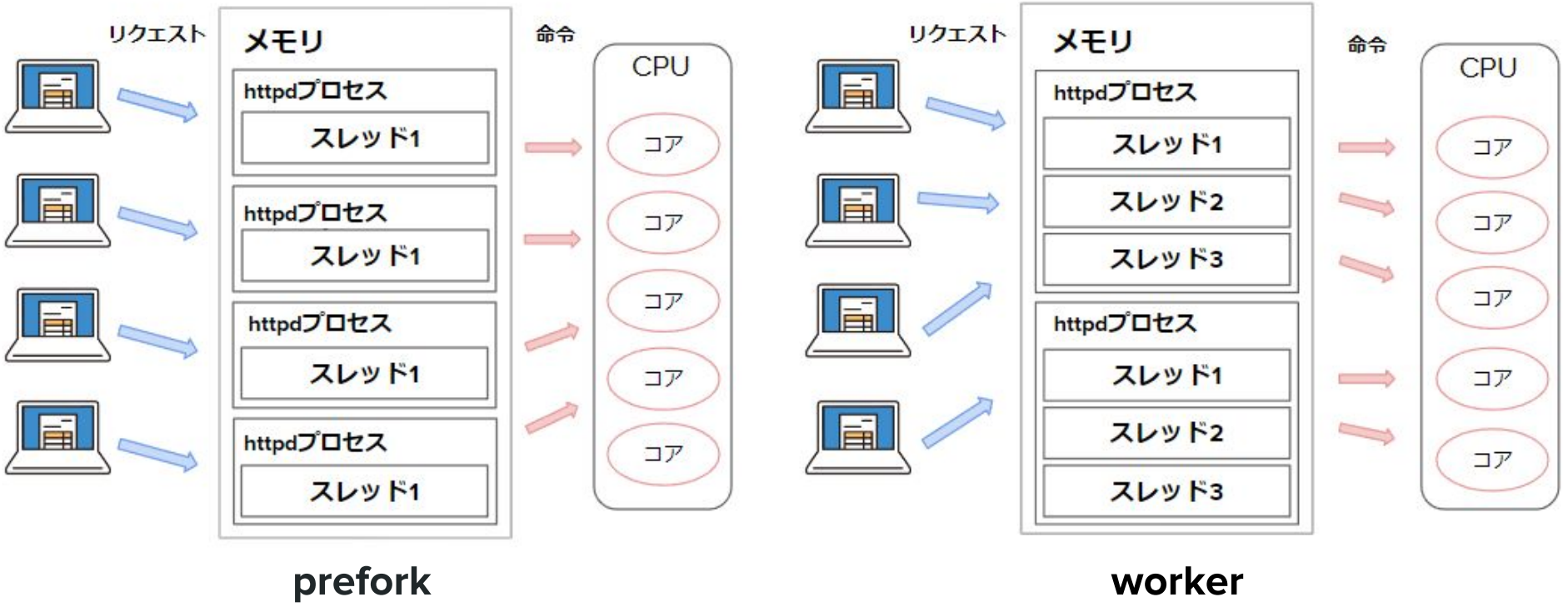
MPMの種類

- **prefork** / **worker** / (event driven)

  当時event drivenはまだ無かった

preforkはプロセスを増やす

イメージ



処理の流れ

リクエスト受け取る

⬇

プロセス / スレッド を増やす

⬇

プロセス数の上限に達する

⬇

新たにプロセスを生成出来ず

あるリクエストの処理が終わるまで新たな接続は待たなければならない

この問題を解決する為...

NGINXの誕生

NGINXのリクエスト処理方法の特徴

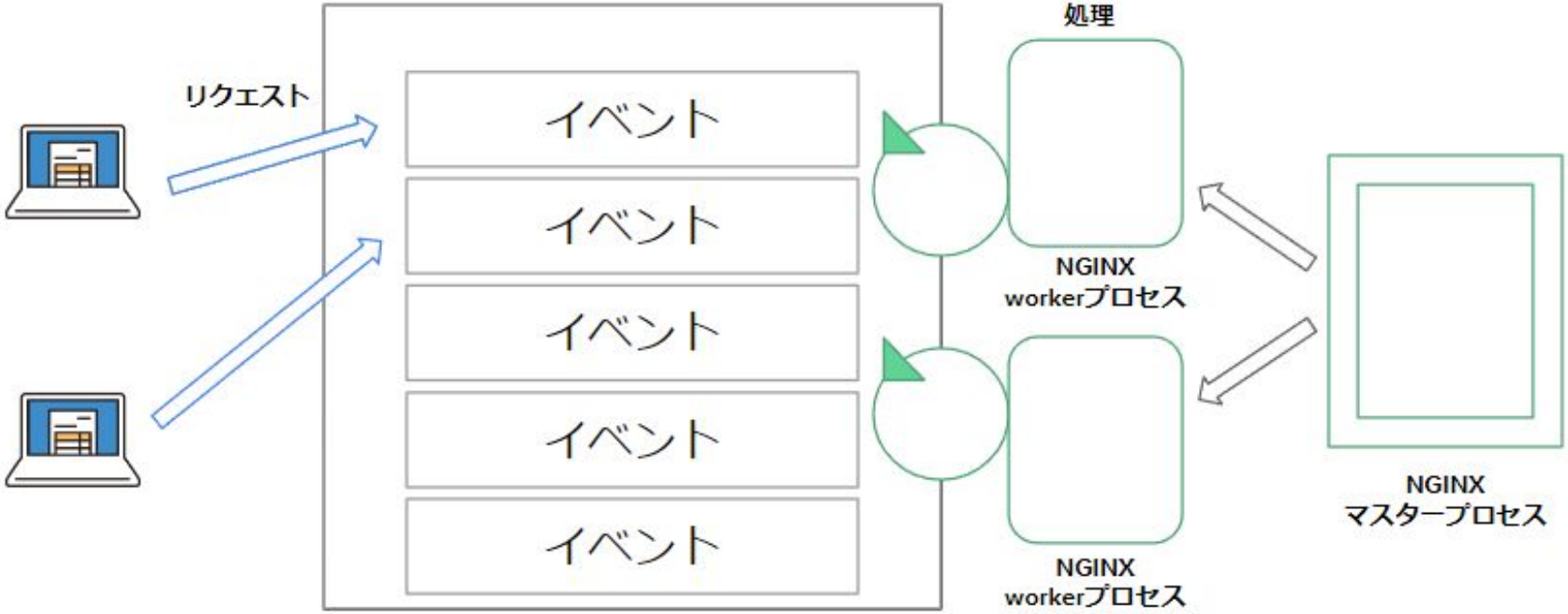
- **イベント駆動**

→クライアントからのリクエストをイベントとして
プロセス内で処理を実行する

- **ノンブロッキングI/O**

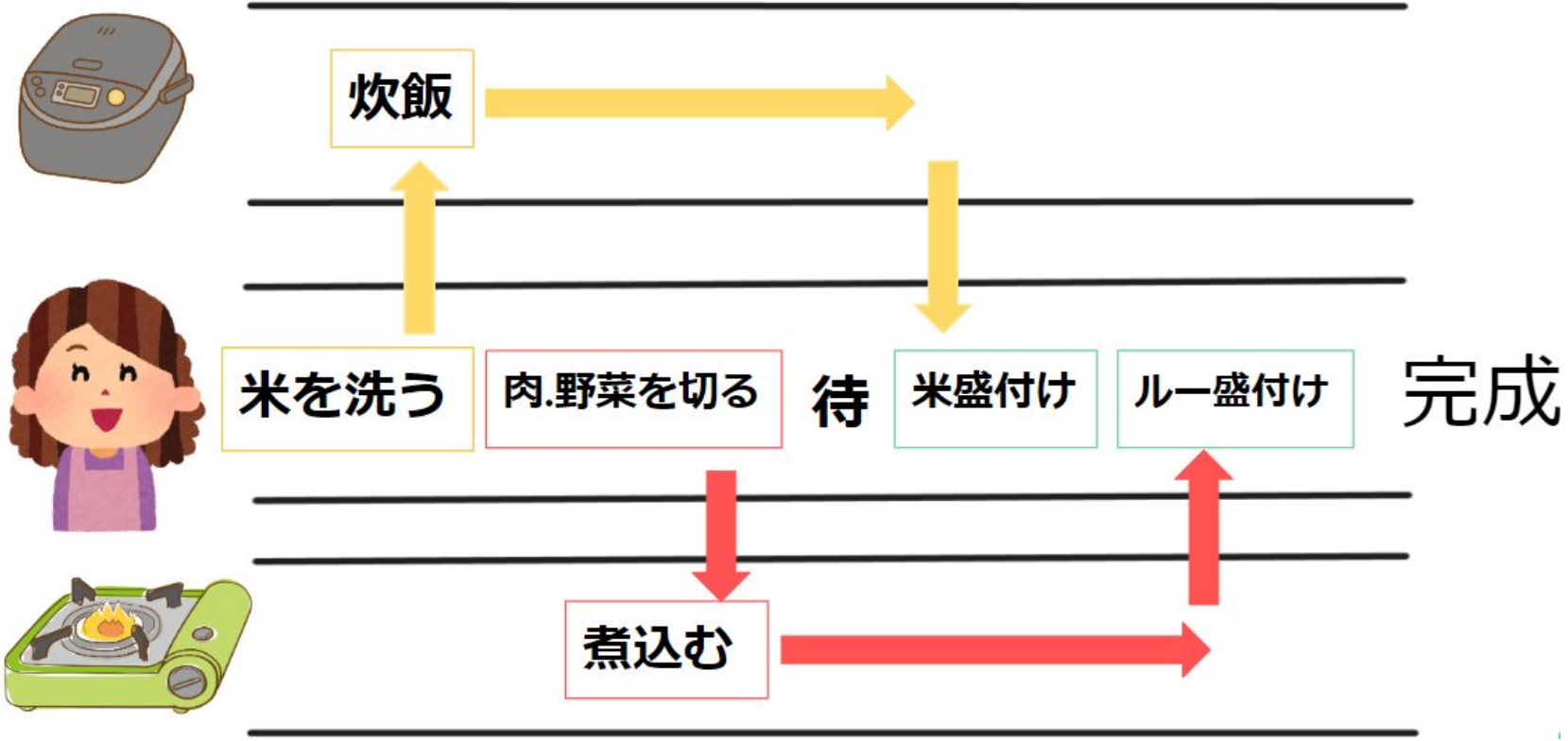
→**データ処理の完了を出来るだけ待たずに**
他の処理を行うこと

イベント駆動のイメージ



ノンブロッキングI/Oのイメージ

一連の処理をカレー作りに例えると...



処理の流れ

リクエスト受け取る

⬇

クライアントの接続要求、リクエストの読み込み、応答等の処理を

イベントとして並列実行する

⬇

プロセス数を増やさない為、クライアントが増えても対応出来る

まとめ

Apache1強

↓

インターネットの普及

↓

C10K問題

↓

Apacheが対応出来ない

↓

NGINX開発

☆だからNGINXは大量アクセスに強い

技術の生まれた歴史を学ぶと
理解が深まるし楽しい！