Nodeについて

Nodeとは

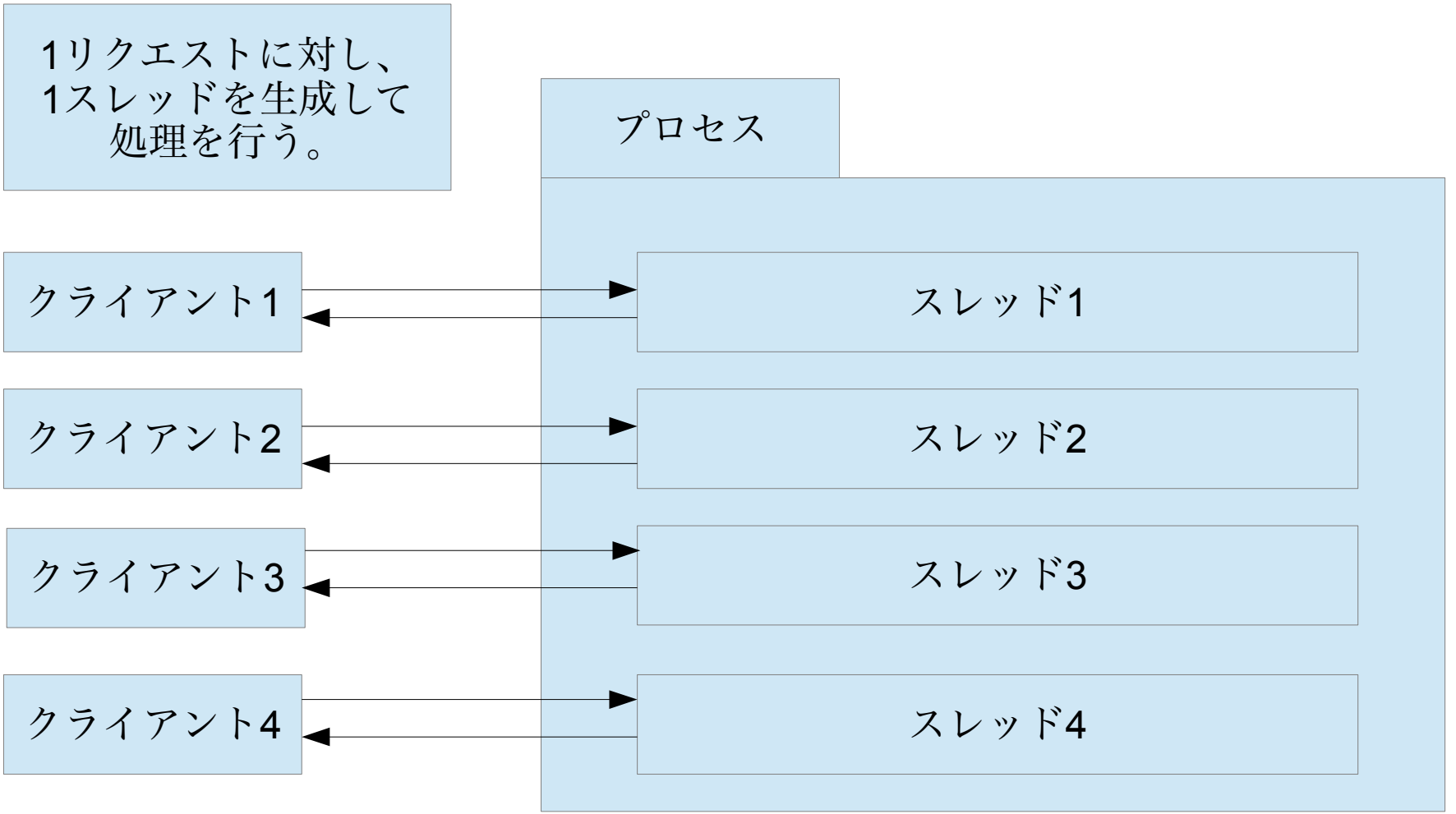
・サーバーサイドJavascriptと呼ばれている技術の一つ。
本家サイト：http://nodejs.org/
日本語コミュニティ： http://nodejs.jp/

・ V8 JavaScript Engineで動作
(Google ChromeのJavaScript Engineと一緒)
https://code.google.com/p/v8/

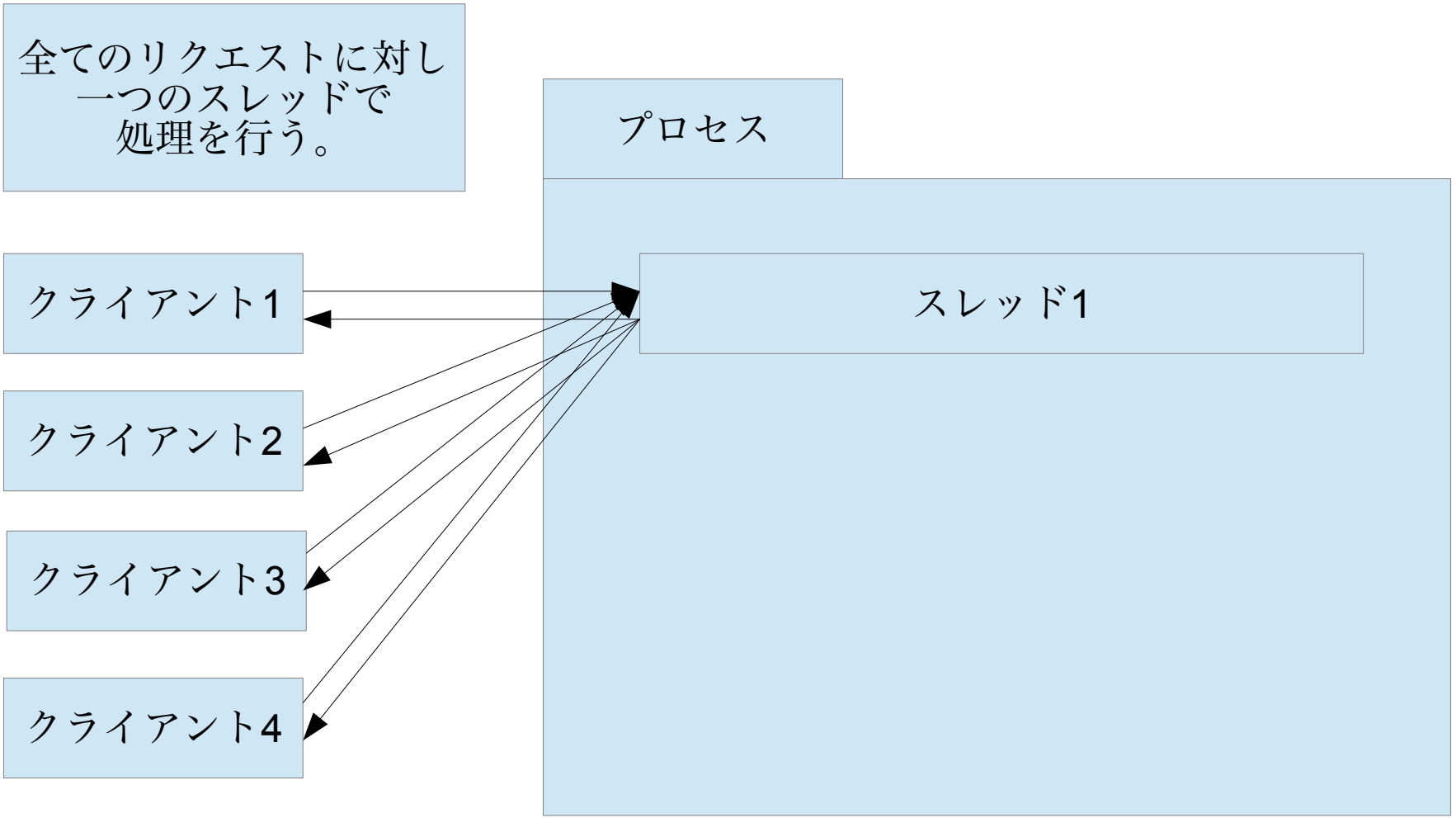
Nodeの特徴

①シングルスレッド
②イベントループ
③ノンブロッキングI/O

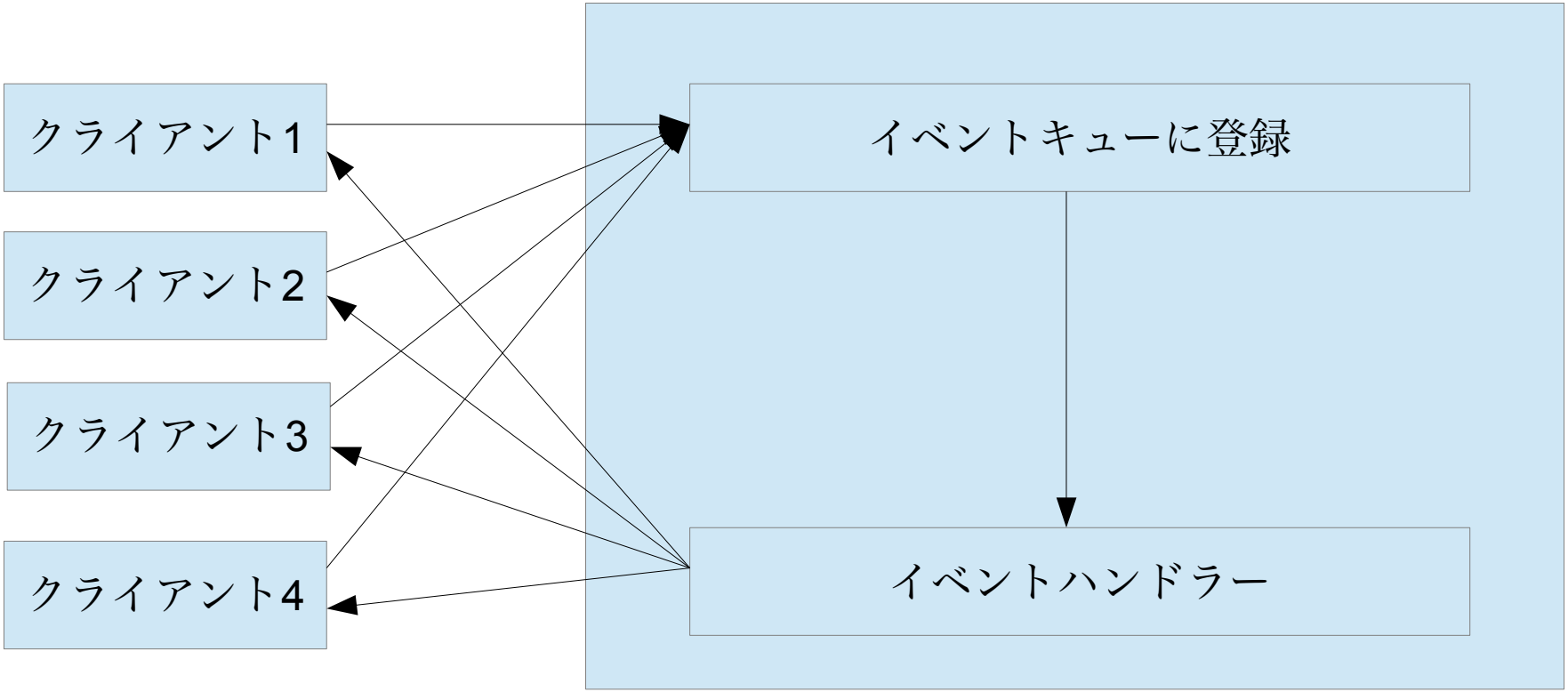
①シングルスレッド
- Apacheの場合



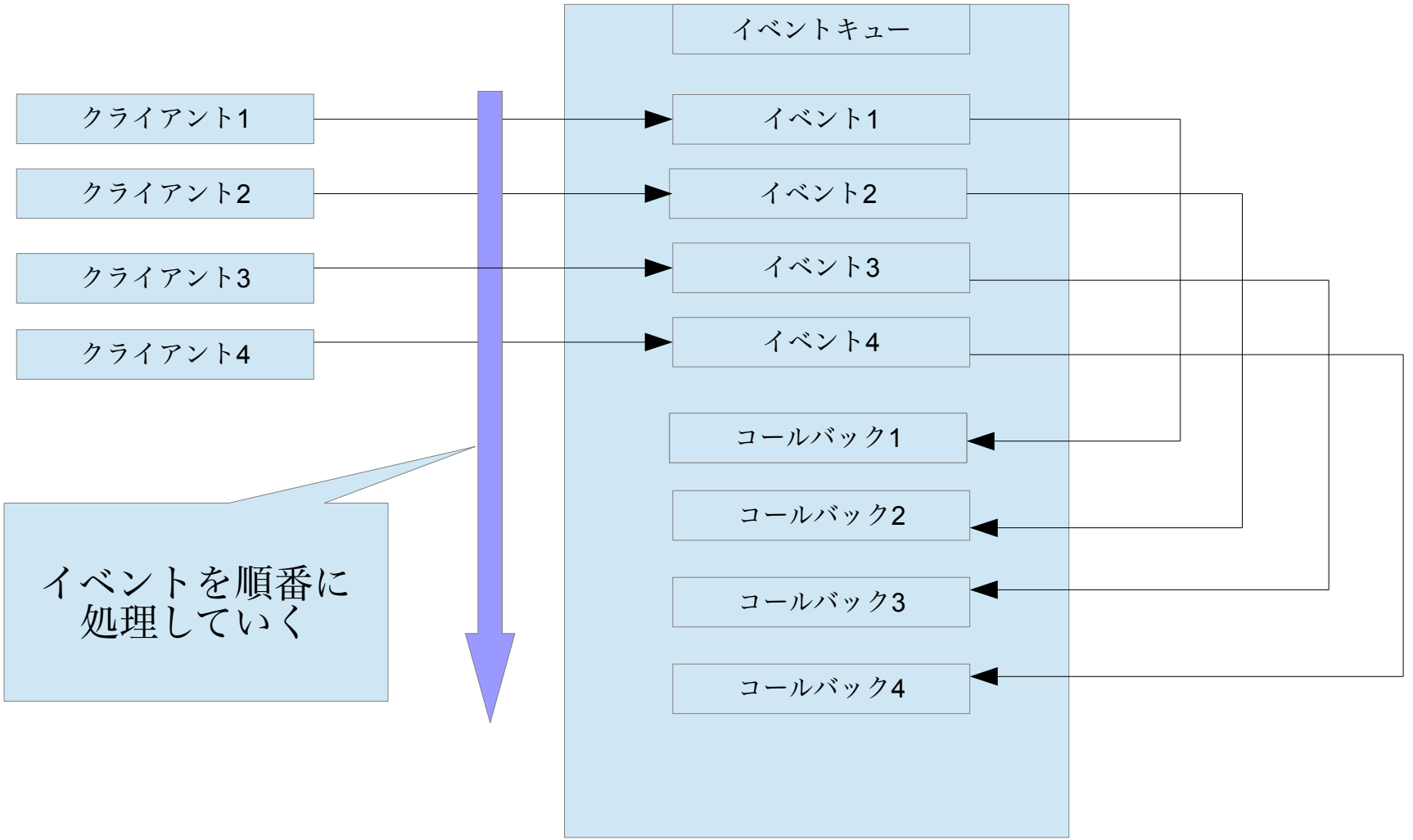
①シングルスレッド - Nodeの場合



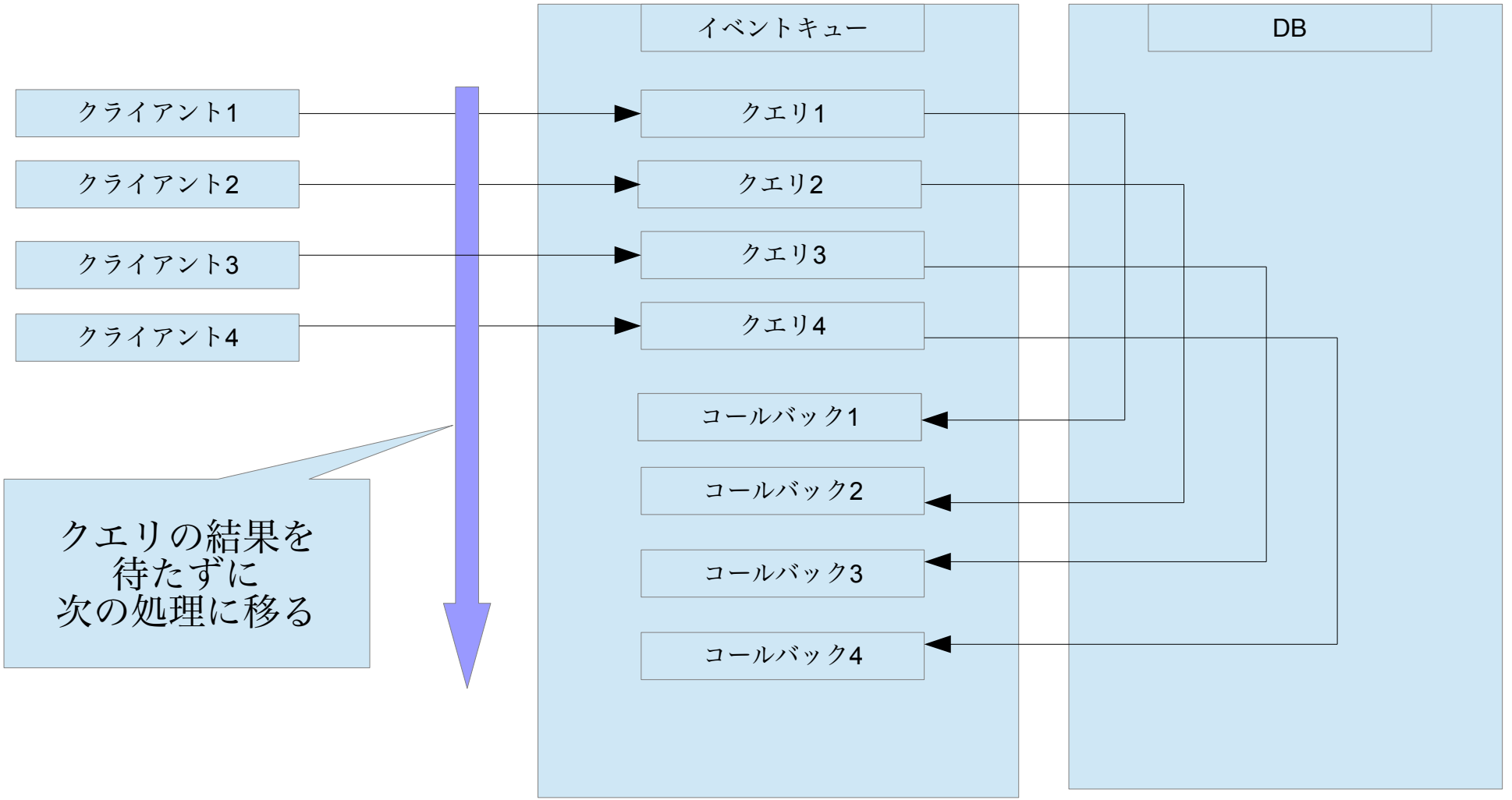
②イベントループ



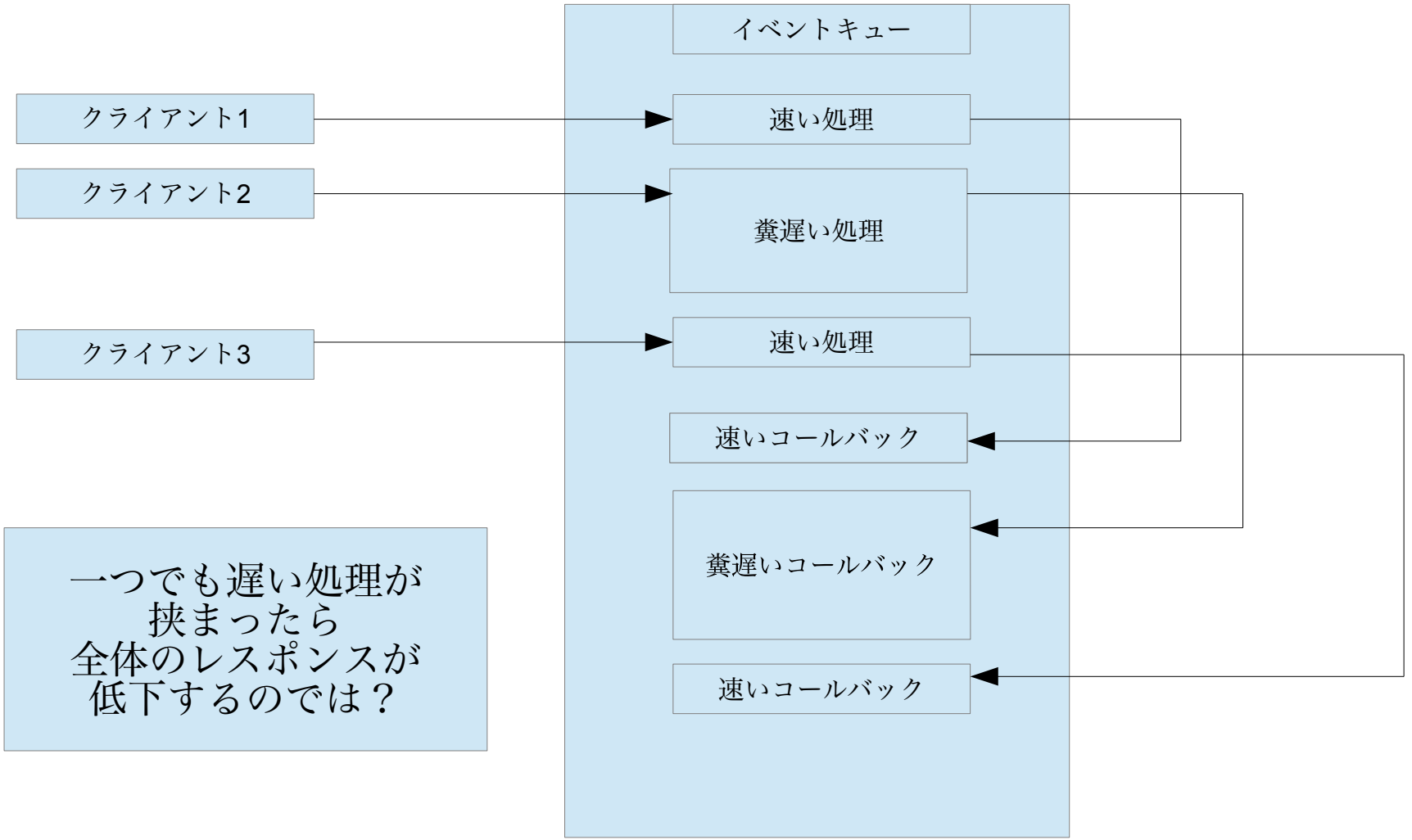
②イベントループ


イベントキュー
クライアント1
クライアント2
クライアント3
クライアント4
イベント1
イベント2
イベント3
イベント4
コールバック1
コールバック2
コールバック3
コールバック4
イベントを順番に
処理していく

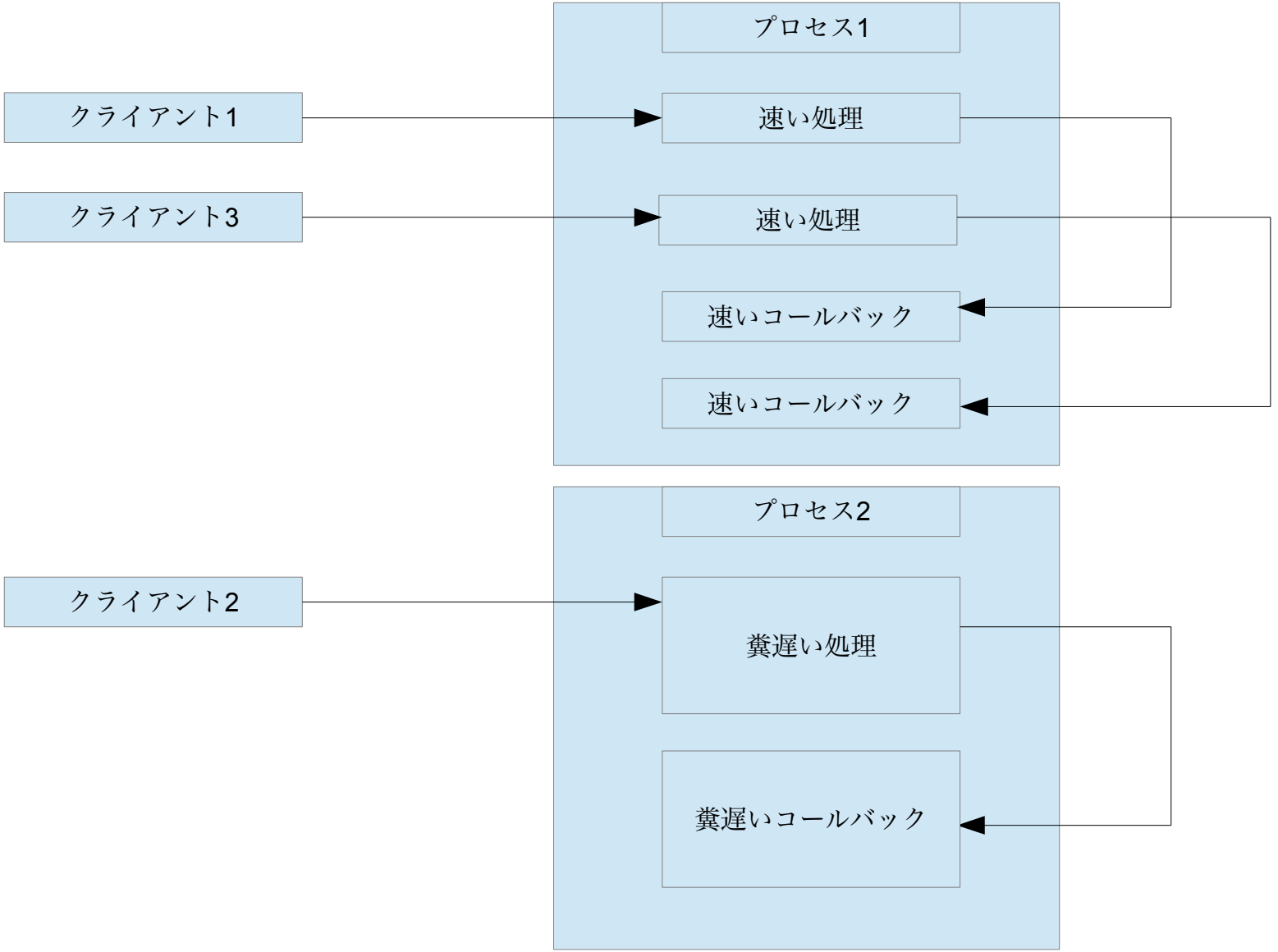
③ノンブロッキングI/O



疑問


イベントキュー
クライアント1
クライアント2
クライアント3
速い処理
糞遅い処理
速い処理
速いコールバック
糞遅いコールバック
速いコールバック
一つでも遅い処理が
挟まったら
全体のレスポンスが
低下するのでは？

別プロセスを生成する
モジュールもある



Nodeの考え方

リクエストが来る度にスレッドを生成していたら
大量のリクエストが来た時の
スレッド生成コストはかなり高くなってしまう。
なので、一つのスレッドで効率的に
リクエストを処理できるようにしよう、
という考え方。

向いている用途(メリット)

中規模程度のwebサイトで、
そんなに時間のかからないリクエストを
大量に処理したい、という時は
軽量なwebサーバーとして簡単に
実装できる。

向いていない用途(デメリット)

・超大規模なwebサイトになってくると、
大量のイベント処理を一つのスレッドで
処理できなくなってくるため、遅くなる。

・時間のかかるリクエスト処理になると
遅くなる。
(一つの処理がスレッドを占有する為)

サンプルコード

たったこれだけ！！

```
var plainHttpServer = http.createServer(function (req, res) {
	res.writeHead(200, { 'Content-Type' : 'text/html' });
	res.end('こんにちわ！！');
}).listen(8880);
```

コンソールで
$node ファイル名
で実行

websocketも簡単に実装できます(クライアントコード)

```
<!DOCTYPE html>
<html>
	<head>
		<title>websocketテスト</title>
		<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
	</head>
	<body>
		<input id="message" type="text"><button id="send">送信する</button>
		<div id="output">ここにデータが表示されます</div>
		<script>
			(function() {
				var ws = new WebSocket("ws:// ご自由に");
				var output = document.getElementById('output');
				var send = document.getElementById('send');

				function logStr(eventStr, msg) {
					return '<div>' + eventStr + ' : ' + msg + '</div>';
				}

				send.addEventListener('click', function() {
					var msg = document.getElementById('message').value;
					ws.send(msg);
					output.innerHTML += logStr('送信', msg);
				});

				ws.onmessage = function (e) {
					console.log(e + "をサーバーから受信。");
					output.innerHTML += logStr('受信', e.data);
				};

				ws.onclose = function (e) {
					output.innerHTML += logStr('切断', e.code + ' - ' + e.type);
				};
			}());
		</script>
	</body>
</html>
```

websocketも簡単に実装できます(サーバコード)

```
var http = require('http');
var WSServer = require('websocket').server;
var clientHtml = require('fs').readFileSync('client.html');

var plainHttpServer = http.createServer(function (req, res) {
	res.writeHead(200, { 'Content-Type' : 'text/html' });
	res.end(clientHtml);
}).listen(8880);

var webSocketServer = new WSServer({ httpServer : plainHttpServer });

webSocketServer.on('request', function (req) {
	req.origin = req.origin || '*';
	var websocket = req.accept(null, req.origin);
	websocket.on('message', function (msg) {
		console.log("" + msg.utf8Data + '"を' + req.origin + 'から受信');
		websocket.send('websocketでこんにちわ！！');
	});

	websocket.on('close', function(code, desc) {
		console.log('接続解除：' + code + ' - ' + desc);
	});
});
```

※npm install websocket で、websocketモジュールをインストールしておくことを忘れずに！！

参考書籍

・Nodeクックブック
http://www.oreilly.co.jp/books/9784873116068/

・はじめてのNode.js -サーバーサイドJavaScriptでWebアプリを開発する
http://www.amazon.co.jp/%E6%9C%AC/dp/4797370904