مختصر كتابي

Who Owns Each Step?



عندما يقول المستخدم إن صفحة الطلب بطيئة، من أين تبدأ البحث؟

جملة «الصفحة بطيئة» تصف ما شعر به المستخدم، ولا تحدد المكوّن البطيء. لذلك نرسم خريطة لـRequest واحد من لحظة الضغط على زر الدفع إلى ظهور النتيجة، ونكتب بجانب كل خطوة السؤال الذي هي مسؤولة عنه، بدل أن نخمّن السبب.

يبدأ **كود الـClient**: يقرأ المنتج وعدد القطع، ويجهّز رسالة الـRequest، ويقرر متى يرسلها وكم ينتظر الرد. إذا أرسل الرسالة مرتين من ضغطة واحدة فسيرى الـServer طلبين، وإذا كانت الرسالة ما زالت عند الـClient فلا ذنب للـDatabase.

إذا كان العنوان اسم موقع، يأتي دور **حل الأسماء** (DNS) ليجد العنوان الشبكي وراء الاسم، وقد تُحفظ النتيجة مدة في أكثر من مكان، فلا تحتاج كل ضغطة إلى Query من الصفر. ثم **النقل**: في الحالة الشائعة يعمل HTTP فوق TCP، فيفتح الطرفان Connection ينقل الـbytes بالترتيب، لكنه لا يعرف أن الرقم داخل الرسالة عدد سماعات، ووصول الـbytes إلى الجهاز ليس دليلًا على أن التطبيق أنهى العمل. وإذا كان العنوان آمنًا، يحمي **TLS** الاتصال ويتحقق من هوية الطرف الآخر: الاسم، وإمكان الوثوق بالـcertificate.

وقد يوجد **Reverse Proxy** قبل التطبيق، يمرر الـRequest إلى نسخة من الخدمة، وقد يفك التشفير ويفتح Connection جديدًا إلى التطبيق، أو يرفض الطلب دون أن يمرره. فلا يوجد سلك واحد من المتصفح إلى الـDatabase؛ لكل وصلة حدودها وسجلاتها ومهلها. ثم يقرأ **التطبيق** الرسالة، ويتحقق من الصلاحية، ويطبّق قواعد العمل، وقد يسأل Database أو خدمة أخرى، فيقضي جزءًا من الوقت في الحساب وجزءًا في الانتظار.

الأزمنة المتداخلة لا تُجمع. قاس الـClient 600 ملي ثانية، وسجّل الـServer 500 داخل التطبيق، منها 400 في انتظار خدمة أخرى: الـ400 جزء من الـ500، والـ500 جزء من الـ600. وإذا وصل المستخدمَ خطأ من الـProxy فلا يعني ذلك أن التطبيق أرسله؛ قارن سجلات الـProxy بسجلات التطبيق. وقد يرى الـServer عنوان الـProxy لا عنوان المتصفح، فلا تعتمد وحدك على header يدّعي عنوان الـClient قبل أن تعرف من وضعه وهل تثق به.

وإذا سجّل الـServer وصول الـRequest بعد 10 milliseconds وانتهاء العمل عند 500، فابدأ الفحص بالمرحلة التي بعد الوصول، لا بحل الاسم؛ هذا لا يحدد السبب، لكنه يضيّق أماكن البحث.

في المعمل المسار أقصر: عنوان محلي رقمي، والـClient يكلّم الـServer مباشرة، بلا DNS خارجي ولا تشفير ولا وسيط ولا Database؛ خدمة واحدة بياناتها في الـMemory.

```
node client.mjs health
node client.mjs path /does-not-exist
```

والـServer يعمل، يعود `404` ومعه `route not found`: النقل نجح والتطبيق نفسه ردّ بأن المسار غير موجود. وبعد إيقاف الـServer يطبع الأمر نفسه `no HTTP response: ECONNREFUSED`، أي لا يوجد من يرد

أصلًا.

- كل خطوة في الرحلة تجيب عن سؤال مختلف، فابدأ بالخريطة لا بالتخمين.
- وصول الـbytes لا يعني أن التطبيق فهم الرسالة أو أنهى العمل.
- الأزمنة المقيسة داخل بعضها لا تُجمع.
- خطأ يصل من الـProxy لا يعني أن التطبيق هو الذي أرسله.
- الرد `404` رد من التطبيق، أما `ECONNREFUSED` فغياب للرد.

جرّب في المعمل تحت الدرس: شغّل `node server.mjs` في الطرفية الأولى، ثم `node client.mjs path` `/does-not-exist` في `Terminal 2`، ثم أوقف الـServer وكرر الأمر وقارن النتيجتين.

الخلاصة: خريطة الرحلة تحوّل شكوى «الصفحة بطيئة» إلى سؤال محدد عن مرحلة لها دليل.