استخدام فصل الأحمال لتجنب الحمل الزائد

ديفيد ياناسيك

عملت لبضع سنوات في فريق أطر عمل الخدمة في Amazon. وصاغ فريقنا الأدوات التي ساعدت مالكي خدمات AWS مثل Amazon Route 53 وElastic Load Balancing على بناء خدماتهم بسرعة أكبر، ويقوم عملاء الخدمة باستدعاء هذه الخدمات بسهولة أكبر. قدمت فرق Amazon الأخرى لمالكي الخدمة وظائف مثل القياس والمصادقة والمراقبة وإنشاء مكتبة العميل وإنشاء الوثائق. بدلاً من اضطرار كل فريق خدمة إلى دمج هذه الميزات في خدماتهم يدويًا، قام فريق أطرات عمل الخدمة بهذا التكامل مرة واحدة وعرض الوظيفة على كل خدمة من خلال التكوين.

وكان أحد التحديات التي واجهناها يكمن في تحديد كيفية توفير الإعدادات الافتراضية المعقولة، خاصةً للميزات المتعلقة بالأداء أو التوفر. فعلى سبيل المثال، لم نتمكن من تعيين مهلة افتراضية من جانب العميل بسهولة، نظرًا لأن إطار عملنا لم يكن لديه أي فكرة عن طبيعة خصائص زمن الاستجابة لمكالمة واجهة برمجة التطبيقات. ولم يكن ذلك سهلاً بالنسبة لمالكي الخدمة أو العملاء لاكتشاف أنفسهم، لذلك واصلنا المحاولة، واكتسبنا بعض الأفكار المفيدة على طول الطريق.

كان من ضمن الأسئلة الشائعة التي واجهناها تحديد العدد الافتراضي للاتصالات التي سيسمح الخادم بفتحها للعملاء في نفس الوقت. تم تصميم هذا الإعداد لمنع خادم من تولي الكثير من العمل ويصبح محملاً أكثر من طاقته الاستيعابية. وبشكل أكثر تحديدًا، أردنا تكوين إعدادات الاتصالات القصوى للخادم بما يتناسب مع الحد الأقصى للاتصالات لموازن الأحمال. كان هذا قبل أيام من Elastic Load Balancing، لذلك كانت موازنات أحمال الأجهزة مستخدمة على نطاق واسع.

وشرعنا في مساعدة مالكي خدمات Amazon وعملاء الخدمة على تحديد القيمة المثالية لأقصى عدد من الاتصالات لتعيين موازن الأحمال، والقيمة المقابلة المراد تعيينها في أطر العمل التي قدمناها. وقررنا أنه إذا استطعنا معرفة كيفية استخدام الحكم البشري لاتخاذ قرار، يمكننا حينئذٍ كتابة برامج لمحاكاة هذا الحكم.

وانتهي الأمر بتحديد القيمة المثالية إلى كونها تمثل تحديًا كبيرًا. عند تعيين الحد الأقصى للاتصالات، فقد يوقف موازن الأحمال الزيادات في عدد الطلبات، حتى عندما يكون للخدمة سعة كبيرة. وعند تعيين الحد الأقصى للاتصالات على عالية جدًا، ستصبح الخوادم بطيئة وغير مستجيبة. وعند تعيين الحد الأقصى للاتصالات مباشرة لحمل العمل، فإن حمل العمل سيتغير أو سيتغير أداء التبعية. حينئذ ستكون القيم خاطئة مرة أخرى، ما يؤدي إلى وجود انقطاعات غير ضرورية أو أحمال زائدة.

وفي النهاية، وجدنا أن الحد الأقصى لمفهوم الاتصالات كان غير دقيق للغاية لتقديم الإجابة الكاملة على هذا اللغز. في هذه المقالة، سنقوم بوصف الطرق الأخرى مثل فصل الأحمال التي وجدناها ناجحة.

تحليل الحمل الزائد

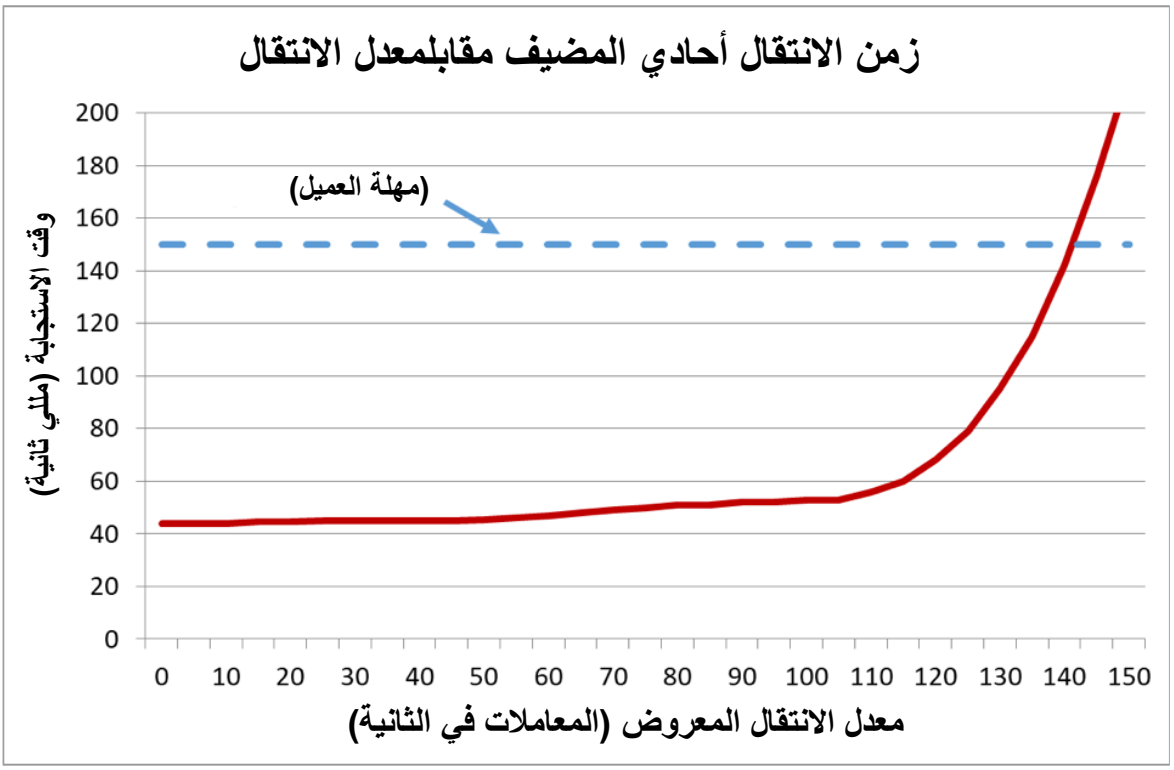
في Amazon، نتجنب الحمل الزائد من خلال تصميم أنظمتنا للتوسع على نحو استباقي، قبل أن تواجه حالات حمل زائد. ومع ذلك، تنطوي أنظمة الحماية على حماية في الطبقات. ويبدأ هذا بالتدرج الآليبالتدرج الآلي، ولكنه يشمل أيضًا آليات فصل الحمل الزائد بأمان، والقدرة على مراقبة تلك الآليات، والأهم من ذلك، الاختبار المستمر.

عندما نقوم باختبار حمل خدماتنا، نجد أن زمن الاستجابة لخادم عند الاستخدام المنخفض أقل من زمن الاستجابة عند الاستخدام العالي. أثناء الحمل الثقيل، تزداد حدة منافسة سلسلة الأنشطة التشغيلية وتبديل المحتوى وجمع المهملات والمنافسة على اتصال الإدخال/الإخراج. في النهاية، تصل الخدمات إلى نقطة الانقلاب حيث يتدهور أداؤها بسرعة أكبر.

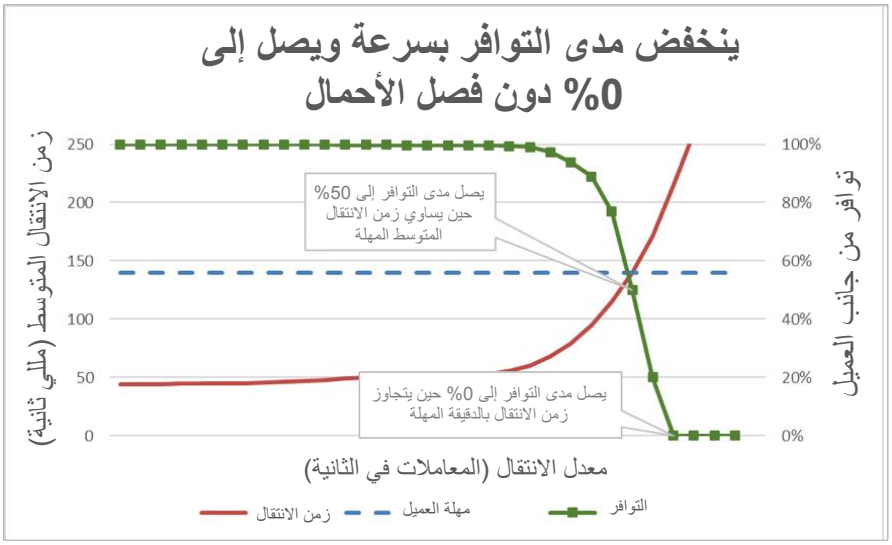
تُعرف النظرية الكامنة وراء هذه الملاحظة باسم قانون القابلية للتوسع الشامل، والذي يعد اشتقاقاً من قانون أمدال. تنص هذه النظرية على أنه بالرغم من إمكانية زيادة إنتاجية النظام باستخدام التوازي، إلا أنه مقيد في النهاية بإنتاجية نقاط التسلسل (أي بالمهام التي لا يمكن توازيها).

لسوء الحظ، لا تتقيد الإنتاجية فقط بموارد النظام، ولكن عادةً ما تقل الإنتاجية عندما يكون النظام محملاً بشكل زائد. عندما يُعطى النظام أعمالاً تفوق دعم موارده، يصبح بطيئًا. تقوم أجهزة الكمبيوتر بالعمل حتى عندما تكون محملة بأكثر من طاقتها، ولكنها تقضي أوقات متزايدة في تبديل المحتوى وتصبح بطيئة للغاية بحيث لا تكون مفيدة.

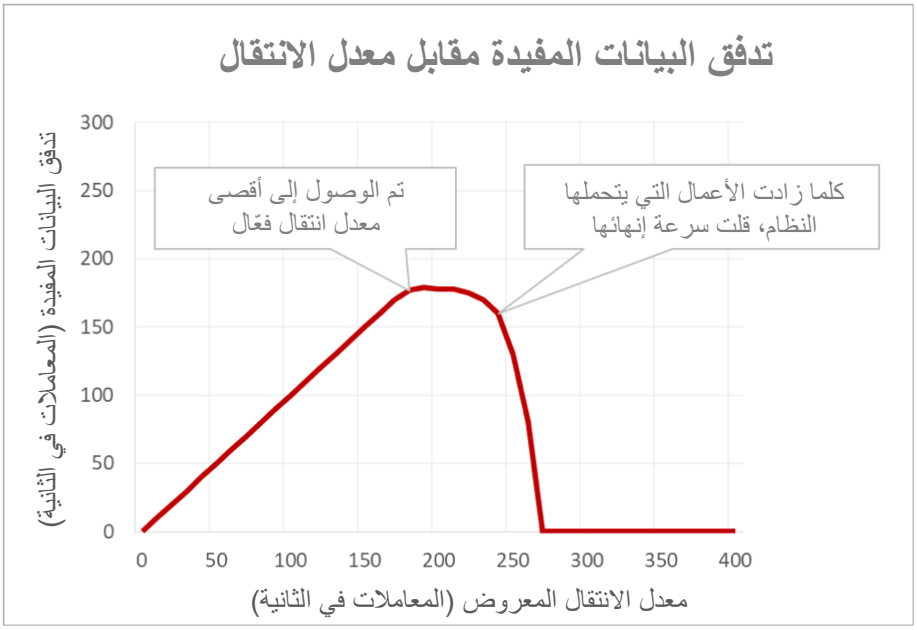
في النظام الموزع الذي يتحدث فيه العميل مع خادم ما، ينفد صبر العميل عادةً ويتوقف عن انتظار استجابة الخادم بعد مرور بعض الوقت. تُعرف هذه المدة باسم *المهلة*. عندما يتم تحميل الخادم بأكثر من اللازم بحيث يتجاوز زمن الانتقال الخاص به المهلة الخاصة بعميله، تبدأ الطلبات بالفشل. يوضح الرسم البياني التالي كيف يزيد وقت استجابة الخادم مع زيادة الإنتاجية المعروضة (بالمعاملات في الثانية)، ويصل وقت الاستجابة في النهاية إلى نقطة الانقلاب حيث تتدهور الأمور بسرعة.



في الرسم البياني السابق، عندما يتجاوز وقت الاستجابة مهلة العميل، يكون من الواضح أن الأمور سيئة، لكن الرسم البياني لا يوضح حجم المشكلة. ولتوضيح ذلك، يمكننا رسم مدى التوفر المتصور للعميل إلى جانب زمن الاستجابة. بدلاً من استخدام قياس وقت استجابة عام، يمكننا استخدام زمن الاستجابة المتوسط، مما يعني أن زمن الاستجابة المتوسط 50 بالمائة من الطلبات كانت أسرع من القيمة المتوسطة. إذا كان زمن الاستجابة المتوسط للخدمة مساويًا لزمن انتظار العميل، فإن نصف الطلبات تنتهي مهلتها، وبالتالي يكون نسبة التوافر 50 بالمائة، وهنا تحول زيادة زمن الاستجابة مشكلة زمن الاستجابة إلى مشكلة توافر. وفيما يلي رسم بياني لما يحدث:



لسوء الحظ، هذا الرسم البياني صعب القراءة. وتتمثل طريقة أبسط لوصف مشكلة التوافر في التمييز بين *تدفق البيانات المفيدة* و*تدفق البيانات الإجمالية*. تمثل الإنتاجية إجمالي عدد الطلبات في الثانية التي يتم إرسالها إلى الخادم. بينما تمثل تدفق البيانات المفيدة مجموعة فرعية من تدفق البيانات الإجمالية التي تتم معالجتها دون أخطاء في زمن استجابة منخفض بما يكفي لاستفادة العميل من الاستجابة.



حلقات الملاحظات الإيجابية

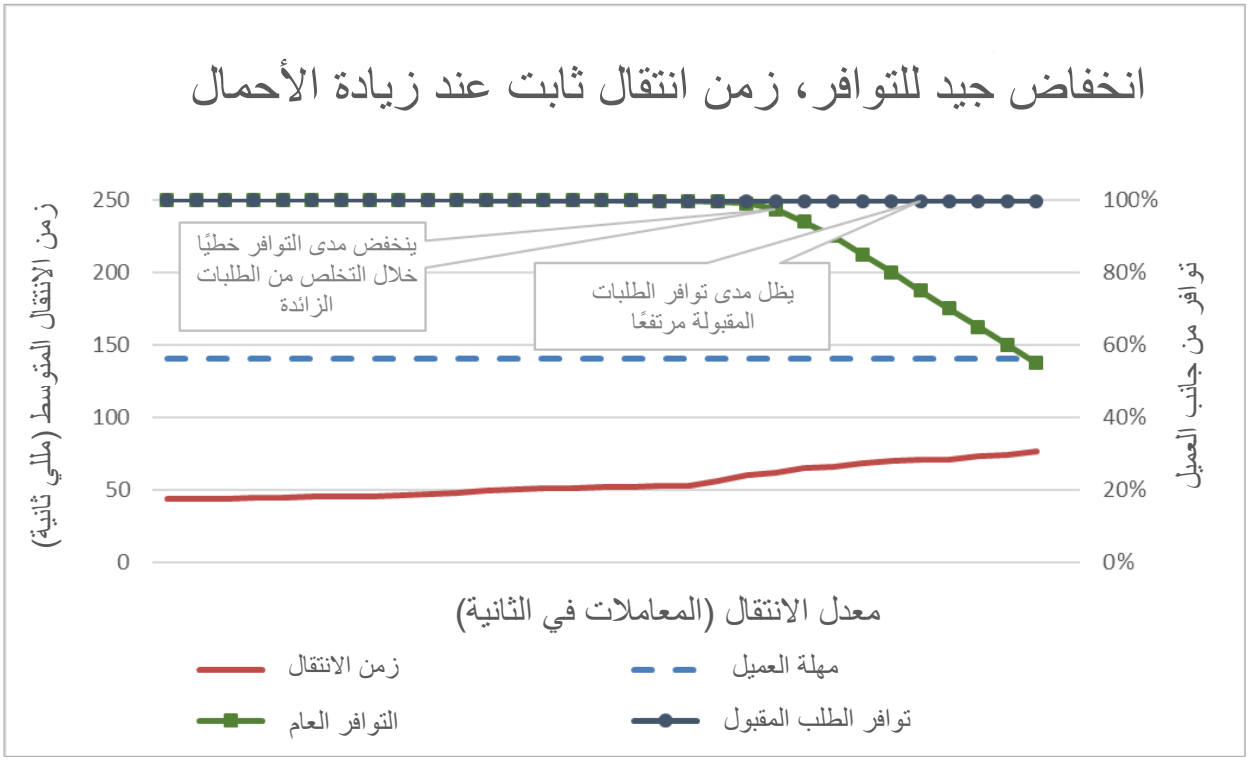
يكمن الجزء الماكر من موقف الحمل الزائد في كيفية تضخيم نفسه في حلقة الملاحظات. عندما تنقضي مهلة العميل، سيسوء الأمر بما يكفي بحصول العميل على خطأ. والأسوأ من ذلك هو أن كل التقدم الذي أحرزه الخادم حتى الآن بشأن هذا الطلب يذهب سدى. والشيء الأخير الذي يجب على النظام القيام به في حالة الحمل الزائد عندما تكون القدرة مقيدة، هو فقدان العمل.

وما يجعل الأمور أسوأ هو أن العملاء في كثير من الأحيان يحاولون تقديم طلباتهم مرة أخرى، مما يضاعف الحمل على النظام. وإذا كان هناك مخطط استدعاء عميق بما يكفي في بنية موجهة نحو الخدمة (أي، يقوم العميل باستدعاء خدمة تستدعي خدمات أخرى، وهي بدورها تستدعي خدمات أخرى) وإذا كانت كل طبقة تقوم بعدد من المحاولات، فإن الحمل الزائد في الطبقة السفلية يؤدي إلى إعادة المحاولات المتتالية التي تضخِّم الحمل المقدم بشكل كبير.

عند اجتماع هذه العوامل، يخلق الحمل الزائد حلقة ملاحظات تؤدي إلى حمل زائد بحالة ثابتة.

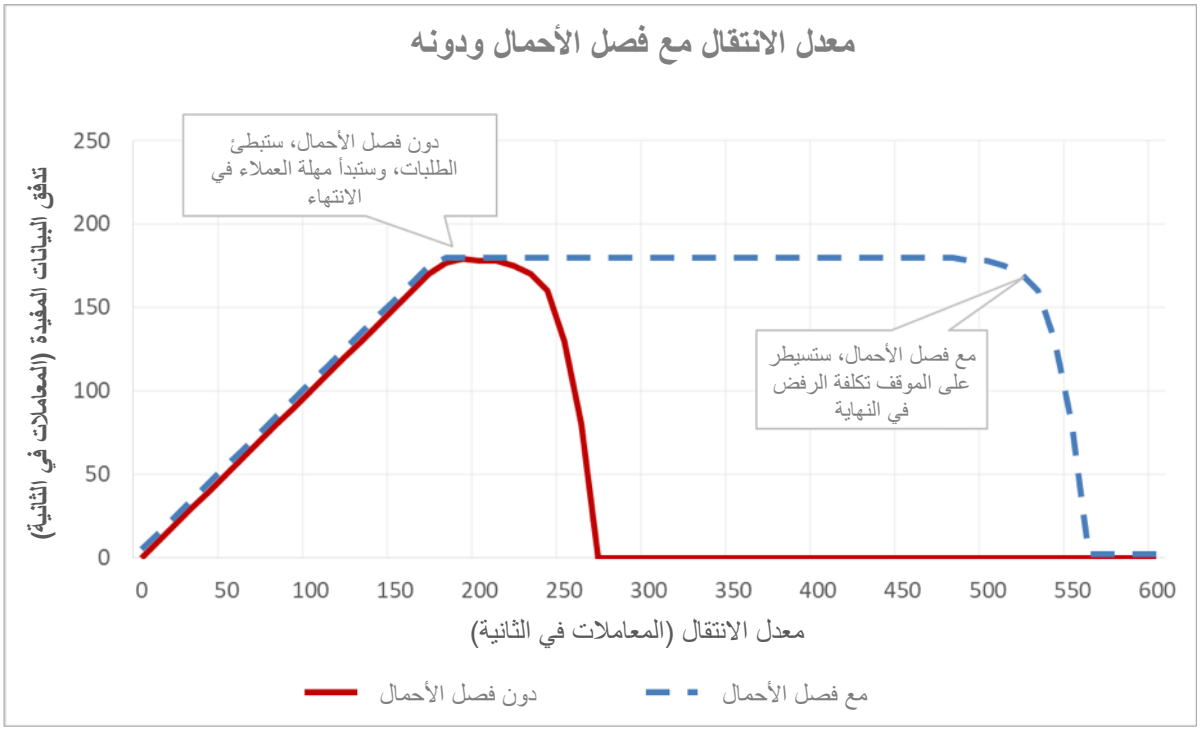
منع العمل من الضياع سدى

يعد فصل الأحمال على السطح بسيطًا. عندما يقترب الخادم من الحمل الزائد، يجب أن يبدأ في رفض الطلبات الزائدة حتى يتمكن من التركيز على الطلبات التي سمح بها. يتمثل الهدف من فصل الأحمال في الحفاظ على تقليل زمن الاستجابة للطلبات التي يقرر الخادم قبولها بحيث تستجيب الخدمة قبل انقضاء مهلة العميل. باستخدام هذا النهج، يحتفظ الخادم بتوفر عالٍ للطلبات التي يقبلها، ويتأثر فقط التوافر بحركة المرور الزائدة.



يساعد الحفاظ على مراقبة زمن الاستجابة عن طريق فصل الحمل الزائد في جعل النظام أكثر توافرًا. لكن يصعب تصور فوائد هذا النهج في الرسم البياني السابق. لا يزال خط التوافر الكلي ينجرف لأسفل، وهو ما يبدو أمرًا سيئًا. ويتمثل العامل الأساسي في أن الطلبات التي قرر الخادم قبولها تظل متوفرة لأنه تم تقديمها بسرعة.

يتيح فصل الأحمال للخادم الحفاظ على تدفق البيانات المفيدة واستكمال أكبر عدد ممكن من الطلبات، حتى مع زيادة تدفق البيانات الإجمالية المقدمة. ومع ذلك، فإن عملية فصل الأحمال ليست بدون مقابل، لذلك في نهاية المطاف يقع الخادم فريسة لقانون أمدال وينخفض معدل تدفق البيانات المفيدة.



الاختبار

عندما أتحدث مع مهندسين آخرين حول فصل الأحمال، أود أن أشير إلى أنه إذا لم يتم اختبار أحمال خدماتهم إلى النقطة التي تنقطع فيها، وبعيدًا عن النقطة التي تنقطع فيها الخدمة، ينبغي عليهم افتراض أن الخدمة ستتعطل بأقل طريقة ممكنة. في Amazon، نقضي وقتًا طويلاً في اختبار خدماتنا. يساعدنا إنشاء الرسوم البيانية، مثل الرسوم الموضحة سابقًا في هذه المقالة، في تحديد خط الأساس لأداء الحمل الزائد وتتبع كيف نتصرف بمرور الوقت أثناء إجراء تغييرات على خدماتنا.

هناك أنواع متعددة من اختبارات الحمل. تقوم بعض اختبارات الحمل بالتأكد من أن الأسطول يتوسع نطاقه تلقائيًا مع زيادة الحمل بينما يستخدم الآخرون حجم أسطول ثابت. في اختبار الحمل الزائد، إذا تدهور توافر الخدمة بسرعة إلى الصفر مع زيادة تدفة البيانات الإجمالية، فإن ذلك يعد إشارة جيدة على أن الخدمة في حاجة إلى آليات إضافية لفصل الأحمال. تتمثل نتيجة اختبار الحمل المثالية في تدفق البيانات المفيدة للوصول إلى مرحلة الاستقرار عند الاقتراب من الاستخدام الكامل للخدمة، وتظل ثابتة حتى عند تطبيق المزيد من تدفق البيانات الإجمالية.

وتساعد أدوات مثل Chaos Monkey في أداء اختبارات هندسة الفوضى على الخدمات. فعلى سبيل المثال، يمكن أن تزيد الحمل على وحدة المعالجة المركزية أو تؤدي إلى فقدان الحزمة لمحاكاة الظروف التي تحدث أثناء الحمل الزائد. ويتمثل أسلوب اختبار آخر نستخدمه في إجراء اختبار توليد الحمل الموجود أو Canary، ودفع الحمل المستمر (بدلاً من زيادة الحمل) نحو بيئة اختبار، ولكن يبدأ في إزالة الخوادم من بيئة الاختبار تلك. يؤدي ذلك إلى زيادة تدفق البيانات الإجمالية المقدمة لكل مثيل، بحيث يمكنه اختبار تدفق البيانات الإجمالية للمثيل. تُعد هذه التقنية المتمثلة في زيادة الحمل بشكل مصطنع عن طريق تقليل أحجام الأسطول مفيدة لاختبار الخدمة بمعزل، ولكنها ليست بديلاً كاملاً عن اختبارات الحمل الكامل. سيؤدي اختبار الحمل الكامل من النهاية للنهاية أيضًا إلى زيادة الحمل على تبعيات تلك الخدمة، مما قد يكشف عن أزمات أداء أخرى.

نحرص أثناء الاختبارات على قياس مدى التوافر المتصور للعميل وزمن الاستجابة بالإضافة إلى التوافر من جانب الخادم وزمن الاستجابة. عندما يبدأ التوافر من جانب العميل في الانخفاض، فإننا ندفع الحمل إلى ما وراء تلك النقطة. إذا كان فصل الأحمال يعمل، فإن تدفق البيانات المفيدة سيبقى ثابت حتى مع زيادة تدفق البيانات الإجمالية المقدمة إلى ما هو أبعد من إمكانيات الخدمة المتدرجة.

يعد اختبار الحمل الزائد أمرًا حاسمًا قبل استكشاف آليات تجنب الحمل الزائد. فكل آلية تقدم تعقيدًا معينًا. فعلى سبيل المثال، ضع في اعتبارك جميع خيارات التكوين في أطر عمل الخدمة التي ذكرتها في بداية المقالة، ومدى صعوبة تصحيح الإعدادات الافتراضية. تضيف كل آلية لتجنب الحمل الزائد حماية مختلفة ولها فعالية محدودة. فمن خلال الاختبار، يمكن للفريق اكتشاف أزامت الأداء بنظامهم وتحديد مجموعة وسائل الحماية التي يحتاجون إليها لمعالجة الحمل الزائد.

الرؤية

في Amazon، نفكر مليا في المقاييس والرؤية التي نحتاجها عند سريان مفعول هذه التدابير المضادة الزائدة بصرف النظر عن التقنيات التي نستخدمها لحماية خدماتنا من الحمل الزائد.

عندما يرفض نظام الحماية من خفض الطاقة أي طلب، فإن هذا الرفض يقلل من توافر الخدمة. وعندما تخطئ الخدمة وترفض طلبًا على الرغم من أن لديها سعة (على سبيل المثال، عندما يكون الحد الأقصى لعدد الاتصالات مضبوطًا على منخفض جدًا)، فإنها تؤدي إلى معدل إيجابي خاطئ. ونسعى جاهدين للحفاظ على معدل إيجابي خاطئ للخدمة عند صفر. إذا وجد فريق أن المعدل الإيجابي الخاطئ الخاصة بالخدمة لديه غير صفري على أساس منتظم، فقد تكون الخدمة مضبوطة بصورة حساسة للغاية، أو يزيد حمل المضيفين الفردين بشكل مستمر وشرعي، وقد يكون هناك مشكلة في تغيير الحجم أو موازنة الحمل. في مثل هذه الحالات، قد يكون لدينا بعض مهام ضبط أداء التطبيقات للقيام بها، أو قد ننتقل إلى أنواع مثيل أكبر يمكنها التعامل مع اختلالات الحمل بأمان أكبر.

وفيما يتعلق بالرؤية، عندما يرفض فصل الأحمال الطلبات، فإننا نتأكد من أن لدينا أدوات مناسبة لمعرفة العميل، وأي عملية كان يتصل بها، وأي معلومات أخرى ستساعدنا في ضبط إجراءات الحماية الخاصة بنا. كما نستخدم أجهزة الإنذار لاكتشاف ما إذا كانت الإجراءات المضادة ترفض أي حجم كبير من حركة المرور. عندما يكون هناك انخفاض في الطاقة، فإن أولويتنا تتمثل في إضافة السعة ومعالجة أزمة الأداء الحالية.

هناك اعتبار آخر دقيق ولكنه مهم حول الرؤية في فصل الأحمال. لقد وجدنا أنه من المهم عدم تلويث مقاييس زمن الاستجابة لخدماتنا بزمن استجابة لطلب معطل. ورغم كل ذلك، يجب أن يكون زمن الاستجابة لطلب فصل الأحمال منخفضًا للغاية مقارنة بالطلبات الأخرى. على سبيل المثال، إذا كان يتم فصل أحمال الخدمة بنسبة 60 في المائة من عدد حركات المرور، فقد يبدو زمن الاستجابة للخدمة مذهلًا إلى حد كبير حتى لو كان زمن الاستجابة الناجح للطلب فظيعًا، نظرًا لعدم الإبلاغ عنه كنتيجة لطلبات الفشل السريع.

يؤثر فصل الأحمال على التدرج الآلي وفشل منطقة توافر الخدمات

وإذا تم تكوينه بشكل خاطئ، يمكن أن يؤدي فصل الأحمال إلى تعطيل التدرج الآلي التفاعلي. انظر إلى المثال التالي: تم تكوين خدمة لتدرج تفاعلي يعتمد على وحدة معالجة مركزية وتم تكوينها أيضًا بخاصية فصل الأحمال لرفض الطلبات على هدف وحدة معالجة مركزية مماثلة. ففي هذه الحالة، سيقلل نظام فصل الأحمال عدد الطلبات لإبقاء حمل وحدة المعالجة المركزية منخفضًا، ولن يتلقى التدرج التفاعلي أبدًا أو يحصل على إشارة مؤجلة لإطلاق مثيلات جديدة.

ونحرص أيضًا على مراعاة منطق فصل الأحمال عند وضع حدود التدرج الآلي لمعالجة حالات فشل منطقة توافر الخدمات. يتم توسيع نطاق الخدمات إلى درجة يمكن أن تصبح فيها منطقة توافر الخدمات غير متاحة مع الحفاظ على أهداف زمن الاستجابة لدينا. غالبًا ما تبحث فرق Amazon عن مقاييس النظام مثل وحدة المعالجة المركزية لتقريب مدى قرب الخدمة من الوصول إلى الحد الأقصى للسعة. ومع ذلك، فقد يقترب الأسطول مع فصل الأحمال

بشكل أكبر إلى النقطة التي يتم عندها رفض الطلبات مما تشير إليه مقاييس النظام، وقد لا يكون لديه السعة الزائدة المتوفرة للتعامل مع فشل منطقة توافر الخدمات. فمن خلال فصل الأحمال، نحتاج إلى التأكد من اختبار انقطاع خدماتنا لفهم سعة أسطولنا وقدرته في أي وقت.

في الواقع، يمكننا أن نستخدم فصل الأحمال لتوفير التكاليف عن طريق تشكيل حركة مرور خارج وقت الذروة وغير حرجة. على سبيل المثال، إذا كان أسطول ما يتعامل مع حركة مرور موقع amazon.com، فقد يقرر أن حركة مرور زاحف البحث لا تستحق أن يتم تحجيمها لتوفير منطقة توافر الخدمات الكاملة. ومع ذلك، نحرص للغاية مع هذا النهج. لا تكلف جميع الطلبات نفس التكلفة، كما أن إثبات أن أي خدمة يجب أن توفر التكرار لمنطقة توافر الخدمات لحركة المرور البشرية ولفصل حركة مرور الزاحف الزائدة في نفس الوقت يتطلب تصميمًا دقيقًا واختبارًا متواصلاً وشراء من الشركة. وإذا كان عملاء الخدمة لا يعلمون أنه يتم تكوين الخدمة بهذه الطريقة، فقد يبدو سلوكها أثناء فشل منطقة توافر الخدمات بمثابة انخفاض كبير في التوافر بدلاً من فصل الأحمال غير الحرجة. لهذا السبب، نحاول في بنية موجهة نحو الخدمة دفع هذا النوع من التشكيل في أقرب وقت ممكن (كما هو الحال في الخدمة التي تتلقى الطلب الأولي من العميل) بدلاً من محاولة اتخاذ قرارات تحديد الأولويات العالمية من خلال الحزمة.

آليات فصل الأحمال

عند مناقشة فصل الأحمال والسيناريوهات التي لا يمكن التنبؤ به، ينبغي أيضًا التركيز على العديد من الظروف *التي يمكن التنبؤ بها* والتي تؤدي إلى انخفاض الطاقة. في Amazon، تحتفظ الخدمات بسعة زائدة كافية لمعالجة إخفاقات منطقة توافر الخدمات دون الحاجة إلى إضافة المزيد من السعة. فهي تستخدم خاصية الحظر لضمان العدالة بين العملاء.

ومع ذلك، وعلى الرغم من هذه الإجراءات الوقائية والممارسات التشغيلية، تتمتع الخدمة بقدر معين من السعة في أي وقت من الأوقات، وبالتالي يمكن أن تصبح محملة بأكثر من طاقتها لعدة أسباب. تتضمن هذه الأسباب تغيرات مفاجئة غير متوقعة في حركة المرور، وفقدان مفاجئ لسعة الأسطول (بسبب عمليات النشر السيئة أو غير ذلك)، والعملاء الذين يتحولون من تقديم طلبات رخيصة (مثل القراءات المخزنة مؤقتًا) إلى طلبات باهظة الثمن (مثل، أخطاء التخزين المؤقت أو الكتابة). عندما تصبح الخدمة محمّلة بأكثر من طاقتها، يجب إنهاء الطلبات التي تلقتها؛ أي يجب أن تحمي الخدمات نفسها من انخفاض الطاقة. سنناقش فيما تبقى من هذا القسم بعض الاعتبارات والتقنيات التي قد استخدمناها على مر السنين لإدارة الحمل الزائد.

فهم تكلفة إسقاط الطلبات

ونتأكد من اختبار حمل خدماتنا إلى *أبعد* من نقطة الوصل لاستقرار تدفق البيانات المفيدة. ويتمثل أحد الأسباب الرئيسية لهذا النهج في التأكد من أنه عند إسقاط الطلبات أثناء فصل الأحمال، تكون تكلفة إسقاط الطلب أقل ما يمكن. لقد رأينا أنه من السهل تفويت بيان سجل عرضي أو إعداد مأخذ توصيل، مما قد يجعل إسقاط طلب أعلى تكلفة من الطبيعي.

وفي حالات نادرة، يمكن أن يكون إسقاط الطلب بسرعة أعلى تكلفة من الاحتفاظ به. في هذه الحالات، نقوم بإبطاء الطلبات المرفوضة لمطابقة (على الأقل) زمن الاستجابة فيما يخص الاستجابات الناجحة. ومع ذلك، فينبغي القيام بذلك عندما تكون تكلفة الاحتفاظ بالطلبات منخفضة قدر الإمكان؛ على سبيل المثال، عندما لا يتم ربط سلسلة أنشطة تشغيلية للتطبيق.

تحديد أولويات الطلبات

عندما يكون الخادم محملًا بشكل زائد، يكون لديه فرصة لترتيب الطلبات الواردة لتقرير الطلبات التي سيتم قبولها والتي سيسقطها. ويكون أهم طلب سيحصل عليه الخادم هو طلب اختبار اتصال من موازن الأحمال. وإذا لم يستجب الخادم لطلبات اختبار الاتصال في الوقت المناسب، فسيتوقف موازن الأحمال عن إرسال طلبات جديدة إلى ذلك الخادم لفترة من الوقت، وسيظل الخادم خاملاً. وفي سيناريو انخفاض الطاقة، فإن آخر شيء نريد القيام به هو تقليل حجم أساطيلنا. إلى جانب طلبات اختبار الاتصال، تختلف خيارات تحديد أولويات الطلبات من خدمة إلى أخرى.

تذكر خدمة الويب التي توفر البيانات لتقديمها إلى amazon.com. قد يكون استدعاء خدمة تدعم عرض صفحة الويب لزاحف فهرس البحث أقل أهمية من الطلب الذي ينشئه شخص. فمن المهم تقديم طلبات الزاحف، ولكن من الناحية المثالية يمكن تحويلها إلى وقت خارج أوقات الذروة. ومع ذلك، فإنه في بيئة معقدة مثل amazon.com حيث يتعاون عدد كبير من الخدمات، إذا كانت الخدمات تستخدم أساليب تحديد أولويات متضاربة، فقد يتأثر توافر النظام بأكمله ويمكن أن يضيع العمل.

يمكن استخدام الأولويات والحظر معًا لتجنب الحدود العليا الصارمة للحظر وفي نفس الوقت حماية الخدمة من الحمل الزائد. في Amazon، في الحالات التي نسمح فيها للعملاء بالتفوق على حدود الحظر التي تم تكوينها، قد يتم إعطاء الأولوية للطلبات الزائدة الواردة من هؤلاء العملاء عن الطلبات الموجودة في نطاق الحصة المحددة من عملاء آخرين. ونقضي وقت وطويلاً في التركيز على خوارزميات الوضع لتقليل احتمال أن تصبح سعة التدفق غير متاحة، ولكن مع مراعاة المفاضلات، فإننا نفضل حمل العمل المتاح المتوقع على حمل العمل غير المتوقع.

مراقبة الساعة

إذا حصلت خدمة ما جزئيًا خلال تقديم طلب ولاحظت أن العميل قد انقضت مهلته، فإنه يمكنها تخطي القيام بما تبقى من العمل وتعطيل الطلب عند تلك النقطة. وإلا، يستمر الخادم في العمل على الطلب، ويكون رده المتأخر مثل إبرة في كوم قش. فمن وجهة نظر الخادم، قد كانت استجابة ناجحة، ولكن من وجهة نظر العميل الذي انقضت مهلته، فقد كان خطأ.

وتتمثل إحدى الطرق لتجنب هذا العمل الضائع في تضمين العملاء تلميحات المهلة في كل طلب، والتي تخبر الخادم بالوقت الذي يرغب في انتظاره. ويمكن للخادم تقييم هذه التلميحات وإسقاط الطلبات المحكوم عليها بالفشل بتكلفة قليلة.

يمكن التعبير عن تلميح المهلة هذا باعتباره وقتًا مطلقًا أو مدة محددة. لسوء الحظ، فإن الخوادم في الأنظمة الموزعة سيئة في الاتفاق على ما هو الوقت الحالي بالضبط. يتم تعويض خدمة Amazon Time Sync Service عن طريق مزامنة ساعات مثيلات Amazon Elastic Compute Cloud (Amazon EC) مع أسطول من الساعات المتوفرة التي تتحكم فيها الأقمار الصناعية والساعاة الذرية في كل منطقة AWS. تعد الساعات المتزامنة بشكل جيد مهمة في Amazon لأغراض التسجيل أيضًا. إن مقارنة ملفي السجل على الخوادم التي لديها ساعات خارج المزامنة يجعل استكشاف الأخطاء وإصلاحها أكثر صعوبة مما هو عليه في البداية.

تتمثل الطريقة الأخرى "لمراقبة الساعة" في قياس المدة على جهاز واحد. تعد الخوادم جيدة في قياس المدد المنقضية محليًا لأنها لا تحتاج إلى الحصول على توافق مع الخوادم الأخرى. ولسوء الحظ، فإن التعبير عن المهلات من حيث المدد له مشاكله أيضًا. أحدها، يجب أن يكون المؤقت الذي تستخدمه رتيبًا وألا يتراجع إلى الخلف عندما يتزامن الخادم مع بروتوكول وقت الشبكة (NTP). وهناك مشكلة أكثر صعوبة تتمثل في أنه لقياس مدة، يجب أن يعرف الخادم متى يبدأ ساعة الإيقاف. في بعض سيناريوهات الحمل الزائد، يمكن أن تصطف وحدات التخزين الضخمة من الطلبات في مخازن بروتوكول التحكم في النقل، لذلك فإنه يحلول الوقت الذي يقرأ فيه الخادم الطلبات من التخزين المؤقت، تكون مهلة العميل قد انقضت بالفعل.

كلما كانت الأنظمة في Amazon تعبر عن تلميحات مهلة العميل، فإننا نحاول تطبيقها بشكل عابر. وفي الأماكن التي تشتمل فيها بنية موجهة نحو الخدمة على قفزات متعددة، فإننا ننشر الموعد النهائي «للوقت المتبقي» بين كل قفزة حتى تتمكن خدمة انتقال البيانات من الخادم في نهاية سلسلة الاستدعاء من إدراك مقدار الوقت المتاح لاستجابتها بحيث تكون مفيدة.

بمجرد أن يعرف الخادم الموعد النهائي للعميل، فهناك سؤال عن مكان إنفاذ الموعد النهائي في تنفيذ الخدمة. إذا كانت هناك خدمة تتضمن قائمة انتظار لطلبات، ننتهز هذه الفرصة لتقييم المهلة بعد إلغاء ترتيب كل طلب. ولكن هذا الأمر لا يزال معقدًا إلى حد كبير، لأننا لا نعرف كم من الوقت سيستغرق الطلب. تحتفظ بعض الأنظمة بتقدير للوقت الذي تستغرقه طلبات واجهة برمجة التطبيقات، وتقوم بإسقاط الطلبات في وقت مبكر إذا تجاوز الموعد النهائي الذي أبلغ عنه العميل تقدير زمن الاستجابة. ومع ذلك، فنادرًا ما تكون الأمور بهذه البساطة. فعلى سبيل المثال، تكون إصابات ذاكرة التخزين المؤقت أسرع من حالات فقدها، ولا يعرف المقيّم ما إذا كانت الإصابة هي التي سبقت أم الفقد. أو قد يتم تقسيم موارد الواجهة الخلفية للخدمة، وتكون بعض الأقسام فقط بطيئة. هناك فرصة كبيرة للذكاء، ولكن من الممكن أيضًا لهذا الذكاء أن يأتي بنتائج عكسية في موقف لا يمكن التنبؤ به.

في تجربتنا، لا يزال تطبيق مهلات العميل على الخادم أفضل من البديل، على الرغم من التعقيدات والمفاضلات. بدلاً من تكديس الطلبات وربما يعمل الخادم على الطلبات التي لم تعد مهمة لأي شخص، فقد وجدنا أنه من المفيد فرض "مدة حياة لكل طلب" والتخلص من الطلبات المحكوم عليها بالفشل.

إنهاء ما تم بدؤه

لا نريد أن يضيع أي عمل مفيد هباء، خاصة في حالة الحمل الزائد، حيث إن تبديد العمل يؤدي إلى إنشاء حلقة ملاحظات إيجابية تزيد من الحمل الزائد، حيث إن العملاء غالبًا ما يحاولون إعادة طلب إذا لم تستجب الخدمة في الوقت المناسب. وعندما يحدث ذلك، يتحول طلب واحد يستهلك الموارد إلى العديد من الطلبات المستهلكة للموارد، مما يضاعف الحمل على الخدمة. عندما تنقضي مهلة العملاء ويقوموا بإعادة المحاولة، فإنهم يتوقفون في كثير من الأحيان عن انتظار الرد على اتصالهم الأول بينما يقومون بتقديم طلب جديد باتصال منفصل. إذا أنهى الخادم الطلب الأول وأصدر رداً، فقد لا يستمع العميل، لأنه ينتظر الآن ردًا من طلب إعادة المحاولة.

وتعد مشكلة العمل الضائع هذه السبب في أننا نحاول تصميم خدمات لأداء *عمل* محدود. في الأماكن التي نعرض فيها واجهة برمجة تطبيقات يمكنها إرجاع قدر كبير من مجموعة البيانات (أو أي قائمة على الإطلاق)، نعرضها على أنها واجهة برمجة تطبيقات تدعم خدمة التصفيح. تعرض واجهات برمجة التطبيقات هذه نتائج جزئية ورمزًا مميزًا يمكن للعميل استخدامه لطلب المزيد من البيانات. لقد وجدنا أنه من الأسهل تقدير الحمل الإضافي على إحدى الخدمات عندما يعالج الخادم طلبًا له حد أعلى لمقدار الذاكرة ووحدة المعالجة المركزية ونطاق الشبكة. من الصعب للغاية إجراء التحكم في القبول عندما لا يكون لدى الخادم أي فكرة عن مدة معالجة الطلب.

تتمثل الفرصة الأكثر دقة لإعطاء الأولوية للطلبات في كيفية استخدام العملاء لواجهة برمجة التطبيقات الخاصة بالخدمة. على سبيل المثال، دعنا نقول أن الخدمة بها واجهات برمجة تطبيقات اثنين: **()start** و**()end**. ولإنهاء عملها، يحتاج العملاء إلى أن يكونوا قادرين على استدعاء واجهتي برمجة التطبيقات كلتاهما. في هذه الحالة، يجب أن تعطي الخدمة الأولوية لطلبات **()end** على طلبات **()start**. إذا أعطيت الأولوية **()start**، فلن يتمكن العملاء من إكمال العمل الذي بدؤوه، مما يؤدي إلى انخفاض الطاقة.

يعد التصفيح مكانًا آخر لمشاهدة العمل الضائع. إذا كان يتعين على العميل تقديم عدة طلبات متسلسلة لترتيب الصفحات من خلال نتائج إحدى الخدمات، ورأى فشلًا بعد الصفحة N-1 وتخلص من النتائج، فإنه يهدر مكالمات خدمة N-2 وأي محاولات إعادة يقوم بتنفيذها على طول الطريق. يشير هذا إلى نفس حالة طلبات **()end**، يجب ترتيب أولوية الطلبات الصفحة الأولى بعد الطلبات في ترقيم الصفحات التالية. كما يؤكد أيضًا سبب تصميمنا للخدمات لأداء عمل مقيد وليس ترقيم الصفحات إلى ما لا نهاية من خلال خدمة يستدعونها أثناء عملية متزامنة.

مراقبة قوائم الانتظار

كما أنه من المفيد أيضًا النظر إلى مدة الطلب عند إدارة قوائم الانتظار الداخلية. تستخدم العديد من بنى الخدمة الحديثة قوائم انتظار في الذاكرة لربط مجموعات سلسة أنشطة تشغيلية بطلبات المعالجة أثناء مراحل العمل المختلفة. ومن المحتمل أن يحتوي إطار عمل خدمة الويب مع منفِّذ على قائمة انتظار مكوّنة أمامه. مع أي خدمة تستند إلى TCP، يحتفظ نظام التشغيل بخازن مؤقت لكل مأخذ، ويمكن أن تحتوي هذه المخازن المؤقتة على كمية كبيرة من الطلبات المحصورة.

عندما نسحب العمل خارج قوائم الانتظار، فإننا ننتهز تلك الفرصة لفحص المدة التي كان عليها العمل في قائمة الانتظار. على الأقل، نحاول تسجيل تلك المدة في مقاييس خدمتنا. بالإضافة إلى تقييد حجم قوائم الانتظار، فقد وجدنا أنه يجب وضع حد أعلى لمقدار الوقت الذي يوضع فيه طلب وارد في قائمة انتظار، ونتخلص منها إذا كانت قديمة جدًا. وهذا يحرر الخادم للعمل على الطلبات الأحدث التي لديها فرصة أكبر للنجاح. كنسخة متطرفة من هذا النهج، نبحث عن طرق لاستخدام قائمة انتظار الوارد أخيرًا صادر أولاً (LIFO) بدلاً من ذلك، إذا كان البروتوكول يدعمها. (HTTP/1.1 لا يدعم توزيع طلبات في مسارات على اتصال معيّن TCP قوائم انتظار الوارد أخيرًا صادر أولاً، لكن HTTP/2 يدعمها بشكل عام).

قد تقوم موازنات الأحمال أيضًا بوضع قائمة انتظار للطلبات أو الاتصالات الواردة عندما تكون الخدمات محمّلة فوق طاقتها، وذلك باستخدام ميزة تسمى قوائم انتظار *التغُّير المفاجئ*. يمكن أن تؤدي قوائم الانتظار هذه إلى انخفاض الطاقة، لأنه عندما يتلقى خادم طلبًا في النهاية، فليس لديه فكرة عن المدة التي قضاها الطلب في قائمة الانتظار. ويتمثل الإعداد الافتراضي الآمن بشكل عام في استخدام تكوين التداعيات، والذي يفشل سريعًا بدلاً من وضع الطلبات الزائدة على قائمة الانتظار. في Amazon، تم تعزيز هذا التعلم في الجيل التالي من خدمة Elastic Load Balancing (ELB). استخدم Classic Load Balancer قائمة انتظار تغير مفاجئ، إلا أن Application Load Balancer يرفض حركة المرور الزائدة. وبغض النظر عن التكوين، تراقب الفرق في Amazon مقاييس موازن الأحمال ذات الصلة، مثل عمق قائمة انتظار التغير المفاجئ أو عدد التداعيات غير المباشرة، لخدماتهم.

في تجربتنا، لا يمكن المبالغة في أهمية مراقبة قوائم الانتظار. غالبًا ما أتفاجأ بالعثور على قوائم انتظار في الذاكرة لم أفكر بديهيًا في البحث عنها، في الأنظمة والمكتبات التي أعتمد عليها. وعندما أقوم بالبحث في أنظمة، أجد أنه من المفيد افتراض وجود قوائم انتظار في مكان ما لا أعرفه حتى الآن. بالطبع ، يوفر اختبار الحمل الزائد معلومات أكثر فائدة من البحث في التعليمات البرمجية، طالما يمكنني الخروج بحالات اختبار واقعية مناسبة.

الحماية من الحمل الزائد في الطبقات السفلى

تتكون الخدمات من عدة طبقات — بدءًا من موازنات الأحمال، ومرورًا بأنظمة التشغيل ذات إمكانيات *netfilter* و*iptables*، وحتى أطر عمل الخدمة، إلى التعليمات البرمجية- وتوفر كل طبقة بعض القدرات لحماية الخدمة.

وغالبًا ما يدعم وكلاء HTTP مثل NGINX ميزة الحد الأقصى للاتصالات (*max_conns*) للحد من عدد الطلبات النشطة أو الاتصالات التي ستنقلها إلى خادم الواجهة الخلفية. قد تكون هذه آلية مفيدة، لكننا تعلمنا استخدامها كملاذ أخير بدلاً من خيار الحماية الافتراضي. فمع الوكلاء، من الصعب تحديد أولوية حركة المرور المهمة، وفي بعض الأحيان يوفر تتبع عدد الطلبات الأولية المحلق معلومات غير دقيقة حول ما إذا كانت الخدمة محملة بشكل زائد بالفعل.

في بداية هذه المقالة، وصفت تحديًا قابلني منذ انضممت لفريق أطرات عمل الخدمة. وكنا نحاول تزويد فرق Amazon بإعداد افتراضي موصى به لأقصى قدر من الاتصالات يمكن تكوينه على موازنات الأحمال. وفي النهاية، اقترحنا أن تقوم الفرق بتعيين أقصى قدر من الاتصالات لموازن الأحمال والوكيل مرتفعًا، والسماح للخادم بتنفيذ خوارزميات فصل الأحمال الأكثر دقة مع المعلومات المحلية. ومع ذلك، كان من المهم أيضًا ألا تتجاوز قيمة الاتصالات القصوى عدد مؤشرات ترابط المستمع أو عمليات المستمع أو واصفات الملفات على الخادم، لذلك كان لدى الخادم الموارد اللازمة للتعامل مع طلبات فحص السلامة الهامة من موازن الأحمال.

تعد ميزات نظام التشغيل للحد من استخدام موارد الخادم قوية ويمكن أن تكون مفيدة في حالات الطوارئ. ونظرًا لأننا نعرف أن الحمل الزائد يمكن أن يحدث، فإننا نتأكد من الاستعداد له باستخدام سجلات التشغيل الصحيحة مع أوامر محددة جاهزة. يمكن للأداة المساعدة *iptables* وضع حد أعلى لعدد الاتصالات التي سيقبلها الخادم، ويمكن أن ترفض الاتصالات الزائدة أكثر بكثير من أي عملية خادم. كما يمكن أيضًا تكوينها باستخدام أدوات تحكم أكثر تطوراً، مثل السماح باتصالات جديدة بمعدل مقيَّد، أو حتى السماح بمعدل اتصال محدود أو حساب لكل عنوان IP مصدر. تعتبر عوامل تصفية IP المصدر قوية، ولكن لا تنطبق على موازنات الأحمال التقليدية. ومع ذلك، يحتفظ ELB Network Load Balancer بعنوان IP المصدر للمستدعي حتى في طبقة نظام التشغيل من خلال الشبكة الافتراضية، ما يجعل قواعد *iptables*، على غرار عوامل تصفية IP المصدر، تعمل حسبما هو متوقع.

تنفيذ الحماية في طبقات

في بعض الحالات، ينفد الخادم من الموارد حتى يقوم برفض الطلبات دون إبطائها. ومع وضع هذا الواقع في الاعتبار، فإننا ننظر إلى جميع القفزات بين الخادم وعملائه لنرى كيف يمكنهم التعاون والمساعدة في التخلص من الحمل الزائد. على سبيل المثال، تتضمن العديد من خدمات AWS خيارات فصل الأحمال بشكل افتراضي. عندما نواجه خدمة مع Amazon API Gateway، يمكننا تكوين الحد الأقصى لمعدل الطلب الذي تقبله أي واجهة برمجة تطبيقات. وعندما يتم مواجهة خدماتنا بواسطة بوابة واجهات برمجة التطبيقات أو Application Load Balancer أو Amazon CloudFront، فإنه يمكننا تكوين AWS WAF لتوزيع حركة المرور الزائدة على عدد من الأبعاد.

تخلق الرؤية توترًا صعبًا. يعد الرفض المبكر مهمًا نظرًا لأنه أرخص مكان لإسقاط حركة المرور الزائدة، ولكنه يكون بتكلفة على الرؤية. ولهذا السبب فإننا نطبق الحماية في طبقات: للسماح لخادم بأخذ أكثر مما يمكنه العمل وإسقاط الفائض، وتسجيل المعلومات الكافية لمعرفة حركة المرور التي يسقطها. ونظرًا لوجود حجم كبير جدًا من حركة المرور التي يمكن أن يخفضها الخادم، فإننا نعتمد على الطبقة الموجودة أمامه لحمايته من أحجام المرور الكبيرة.

التفكير في الحمل الزائد بشكل مختلف

ناقشنا في هذا المقال كيف تأتي الحاجة إلى التخلص من الحمل من كون أن الأنظمة تصبح أبطأ نظرًا لتزويدها بمزيد من الأعمال المتزامنة، حيث تساهم قوى مثل حدود الموارد والتنافس في ذلك. يتم دفع حلقة الملاحظات للحمل الزائد من خلال الاستجابة، مما يؤدي في النهاية إلى هدر العمل،

وتضخيم معدل الطلب، وفرض الحمل الزائد. ينبغي تنحية هذه القوة، المدفوعة بقانون قابلية التوسع الشامل وقانون أمدال، عن طريق **التخلص من الحمل الزائد والحفاظ على أداء ثابت يمكن التنبؤ به في مواجهة الحمل الزائد**. ويُعد التركيز على الأداء المتسق والمتوقع مبدأً رئيسيًا للتصميم تُبنى عليه الخدمات في Amazon.

على سبيل المثال، تعتبر Amazon DynamoDB خدمة قاعدة بيانات توفر أداءً وتوافرًا يمكن التنبؤ بهما على نطاق واسع. حتى إذا زاد حمل العمل بسرعة وتجاوز الموارد المتاحة، تحتفظ DynamoDB بزمن استجابة لتدفق البيانات المفيدة يمكن التنبؤ بها لحجم العمل هذا. وتتفاعل العوامل، مثل التدرج الآلي لـDynamoDB ، والسعة المتغيرة،وتتفاعل بسرعةعند الطلب لزيادة معدلات تدفق البيانات المفيدة للتكيف مع الزيادة في حمل العمل. خلال ذلك الوقت، يظل معدل تدفق البيانات المفيدة ثابتًا، مع الحفاظ على الخدمة في الطبقات فوق DynamoDB مع أداء يمكن التنبؤ به أيضًا، وتحسين استقرار النظام بأكمله.

يوفر AWS Lambda مثالًا أوسع حتى للتركيز على الأداء الذي يمكن التنبؤ به. عندما نستخدم Lambda لتنفيذ خدمة ما، يتم تشغيل جميع خدمات استدعاء واجهة برمجة تطبيقات في بيئة التنفيذ الخاصة بها بكميات متسقة من الموارد المخصصة لها، وأن بيئة التنفيذ تعمل على طلب واحد فقط في وقت واحد، وهذا يختلف عن نموذج معتمد على الخادم، حيث يعمل خادم معين على واجهات برمجة تطبيقات متعددة.

إن عزل كل خدمة استدعاء لواجهة برمجة التطبيقات لمواردها المستقلة (حساب، ذاكرة، قرص، شبكة) سيتحايل على قانون أمدال في بعض النواحي، لأن موارد خدمة استدعاء واحدة لواجهة برمجة التطبيقات لن تتعارض مع موارد خدمة استدعاء أخرى. لذلك، إذا تجاوز تدفق البيانات الإجمالية تدفق البيانات المفيدة، فسيظل تدفق البيانات المفيدة مسطحة بدلاً من الانخفاض كما هو الحال في بيئة أكثر تقليدية تعتمد على الخادم. وهذا ليس حلا سحريًا، لأن التبعيات يمكن أن تبطئ وتؤدي إلى ارتفاع التزامن. ومع ذلك، لن يتم تطبيق أنواع التنافس في هذا السيناريو على الموارد المحملة على المضيف التي ناقشناها في هذا المقال على الأقل.

يعد فصل الموارد ميزة بسيطة ولكنها مهمة لبيئات حسابية حديثة دون خوادم مثل AWS Fargate، وAmazon Elastic Container Service (Amazon ECS)، و AWS Lambda. في Amazon، وجدنا أن الأمر يتطلب الكثير من العمل لتنفيذ فصل الأحمال، بدءًا من توليف مجموعات سلسة أنشطة تشغيلية، إلى اختيار التكوين المثالي لاتصالات القصوى لموازن الأحمال. ومن الصعب أو المستحيل العثور على افتراضات معقولة لهذه الأنواع من التكوينات، لأنها تعتمد على الخصائص التشغيلية الفريدة لكل نظام. توفر هذه البيئات الحسابية بدون خادم الأحدث عملية عزل للموارد على مستوى أقل وتكشف عن أزرار ذات مستوى أعلى، مثل عناصر التحكم في الحظر والتزامن، للحماية من الحمل الزائد. وفي بعض الطرق يمكننا تخطي هذا التكوين تمامًا والحماية من فئات الحمل الزائد دون أي تكوين على الإطلاق، بدلاً من البحث عن قيمة التكوين الافتراضية المثالية.

مزيد من القراءة

- قانون القابلية للتوسع الشامل
- قانون أمدال
- تصميم موجه إلى أحداث على مراحل (SEDA)
- قانون ليتل (يصف التزامن في النظام وكيفية تحديد سعة الأنظمة الموزعة)
- سرد قصص عن قانون ليتل، *مدونة مارك*
- نظرة متعمق على Elastic Load Balancing وأفضل الممارسات، عرض تقديمي يتناول فعالية re:Invent 2016 (يصف تطور Elastic Load Balancing لإيقاف إدراج الطلبات الزائدة في قوائم الانتظار)
- Burgess, Thinking in Promises: Designing Systems for Cooperation, O'Reilly Media, 2015