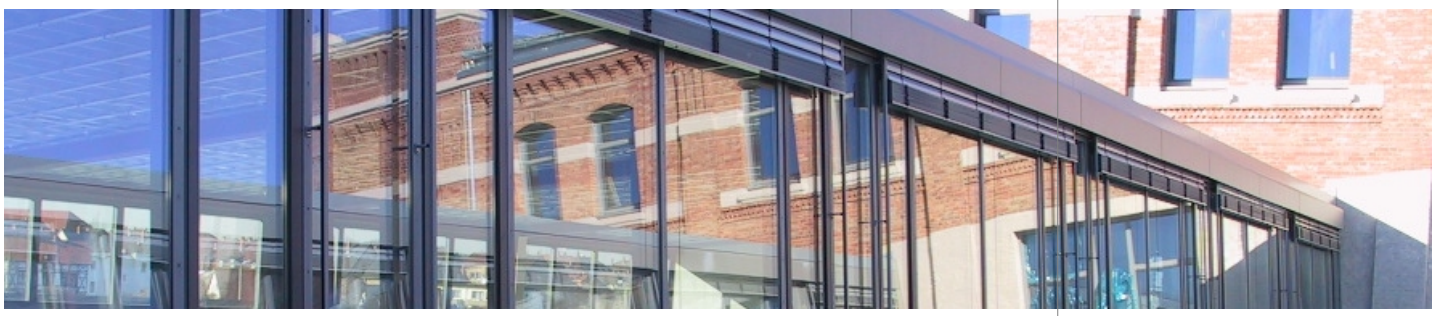
November 2015

yet another newsletter

yasc Informatik GmbH | yasc Coaching GmbH | yasc Consulting GmbH



Anwendungs-Server mit Qt

Einführung

Das Qt Framework ist für die Erstellung von graphischen Benutzeroberflächen in C++ seit vielen Jahren etabliert. Eingabemasken lassen sich schnell und komfortabel erstellen, die Verknüpfung mit der eigentlichen Anwendungsfunktionalität lässt sich über den Qt-eigenen Signal-Slot-Mechanismus elegant und wartungsfreundlich realisieren.

Um das eigentliche GUI-Framework ist im Laufe der Zeit eine reichhaltige Sammlung an Unterstützungsklassen gewachsen. So bringt Qt eine betriebssystemunabhängige Kapselung für Low-Level-Funktionalität mit, z. B. für Threading, Speicherverwaltung, Kollektionen, Marshalling, Datei-, Netzwerk- oder Datenbank-Zugriffe, bis hin zu High-Level-Funktionen wie HTML-Renderer.

All diese Komponenten sind über gemeinsam genutzte Datentypen und Konzepte zu einer dichten, interoperablen Infrastruktur verwoben. Zunehmend finden sich hierin auch viele Komponenten, die Qt über die Rolle eines GUI-Toolkits herauswachsen lassen, z. B. Server-Komponenten für HTTP Web-Sockets, Java-Script-Interpreter oder JSON-Parser.

Bei einem konkreten Kundenprojekt galt es, einen Front-End-Server für ein Geschäftssystem zu entwerfen. Das System sollte in einem mittleren Unternehmen verschiedene Geschäftsvorgänge vor der Auftragserfassung bis zur Übergabe der Rechnungsdaten an ein SAP-System begleiten.

Als Design-Parameter ist von 500 simultan aktiven Nutzern auszugehen, bei bis zu 1500 gleichzeitig bestehenden Verbindungen zum System. Die meisten Nutzer führen nur gelegentlich Operationen auf dem System aus, um ihre Arbeitsschritte zu protokollieren oder Dokumente zu archivieren. Als Verbindung zu den Clients sollte das HTTP-Protokoll verwendet werden.

Aufgrund der überschaubaren Anzahl an Nutzern sollte ein einzelner Server eingesetzt werden. Nutzer-Anfragen sollten eine verlässliche, kurze Antwortzeit vom System erfahren und die Rechnerressourcen sollten fair zwischen den Nutzern verteilt werden. Hierbei ist zu beachten, dass einige Vorgänge im Back-End längere Zeit blockieren können, z. B. die Erstellung von Reports oder die SAP-Kommunikation.

über yasc

Die yasc Unternehmensgruppe ist IT-Dienstleister für Konzerne, Forschungseinrichtungen und mittelständische Unternehmen.

Seit 1997 realisieren wir termingerecht und im Budget Software nach Kundenanforderungen und beraten Unternehmen beim Einsatz von Informationstechnologie.

IT-Projekte im Umfang von einem Monat bis zu zehn Personenjahren führen wir von der Anforderungsanalyse bis zur Systemeinführung und Wartung.

Gern erfüllen wir auch Ihre Anforderungen.

Kontakt

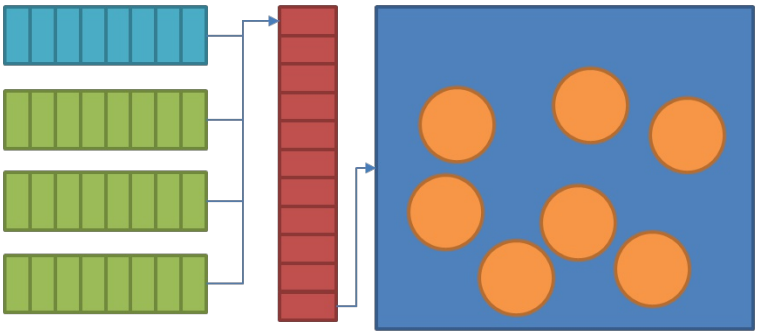
yasc Unternehmensgruppe
Frankfurter Straße 2
38122 Braunschweig

Telefon 0531 / 250 39 39
Fax 0531 / 250 39 41
E-Mail company@yasc.de
Home www.yasc.de

Threading-Modelle

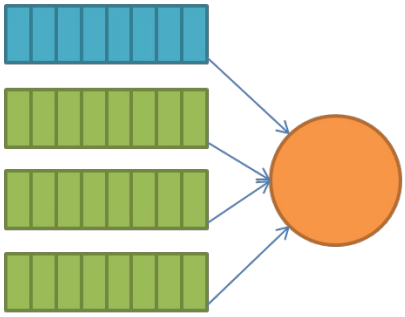
Um diese Anforderungen optimal erfüllen zu können, sollte eine geeignete Thread-Architektur gewählt werden. Hierbei müssen auch die Besonderheiten der Qt-Abstraktionen für Netzwerk-Zugriffe beachtet werden.

Eine effiziente Standard-Architektur für Server auf modernen Multi-Core-Prozessoren ist der Einsatz eines Thread-Pools zur Abarbeitung eingehender Nutzer-Anfragen (Abbildung 1: moderne Architektur). Das Scheduling der Anfragen auf die Threads bleibt dabei dem Betriebssystem überlassen (z. B. durch Einsatz von eselect). Dies minimiert erheblich den Synchronisations-Overhead. Es lässt sich leider innerhalb des Qt-Frameworks nicht implementieren, da die Qt-eigenen Socket-Verbindungen während Ihrer Lebenszeit fix an einen einzelnen Thread gebunden sein müssen.



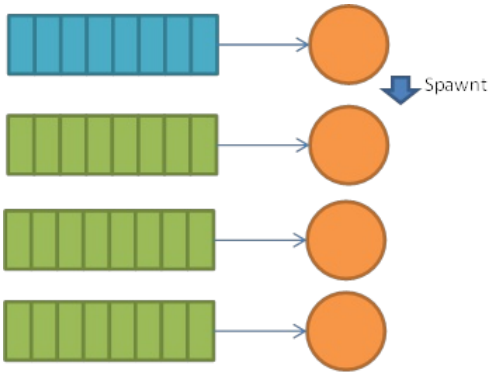
- Abbildung 1: Moderne Architektur -

Die normale Qt-Implementierung eines TCP/IP-Servers sieht einen einzelnen Thread vor, in dem alle Verbindungen und der Server-Socket selbst mittels asynchroner I/O bedient werden (Abbildung 2: Qt-Standard). Während asynchrone I/O selbst eine sehr gute und effiziente Methodik zur Kommunikation mit dem Betriebssystem ist, kann hier keine Lastverteilung in einem Multi-Core-System erreicht werden.
Im Vorliegenden Einsatz-Szenario kommt es zu einem weiteren Problem: Während der Server eine einzelne Abfrage abarbeitet, wäre er für alle anderen Nutzer blockiert. Das ist insbesondere bei im Backend blockierenden Nutzeranfragen nicht akzeptabel.



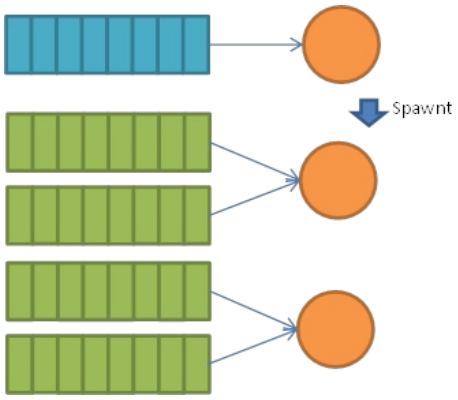
- Abbildung 2: Qt-Standard -

Als Lösung dieses Problems kann der Qt-Socket Server so überschrieben werden, dass ein eigener Thread für jede Verbindung angelegt wird. Der Preis dafür ist allerdings ein erheblicher Overhead für die Verwaltung und Synchronisierung der Threads, der bei den angesetzten 1500 gleichzeitigen Verbindungen nicht mehr akzeptabel ist.



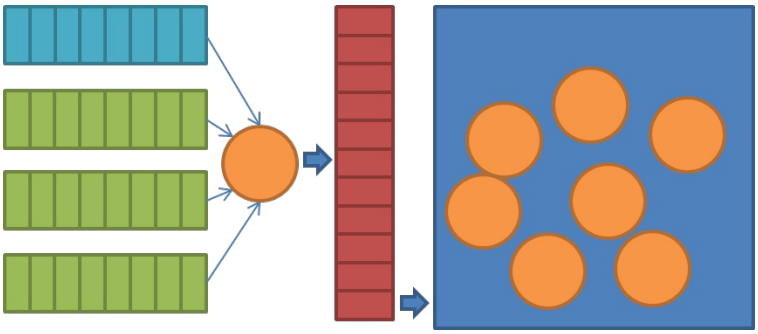
- Abbildung 3: Ein Thread pro Verbindung -

Als Kompromiss können mehrere Verbindungen jeweils einem Thread zugeordnet werden (Abbildung 4: Mischform). Leider führt dies zu einem unvorhersehbaren, möglicherweise unfairem Antwortverhalten des Servers, je nachdem wie sich die Last der Verbindungen auf die einzelnen Threads verteilt.



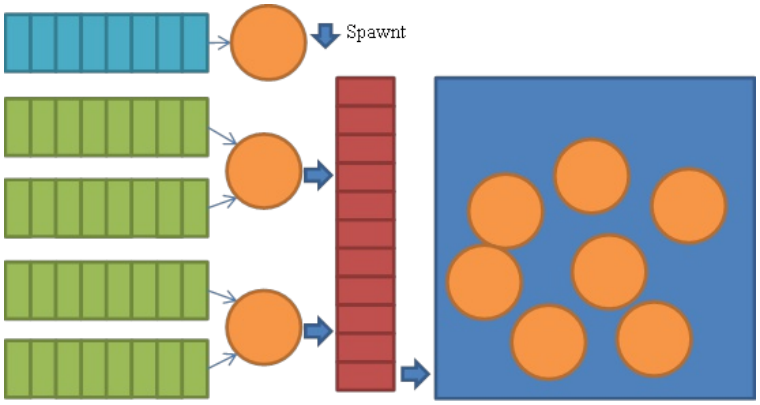
- Abbildung 4: Mischform -

Um der Qt-Besonderheit zu begegnen, dass eine Verbindung nur fest einem Thread zugeordnet werden darf, kann ein Frontend-Thread vorgeschaltet werden (Abbildung 5: Teilung in Front- und Backend). Dieser terminiert die Netzwerk-Verbindung und reicht Anfragen an eine interne Queue durch, aus der sich ein Pool von Backend-Threads bedient. Der Preis für diese Teilung ist allerdings ein erheblicher Synchronisations-aufwand zwischen Front- und Backend. Außerdem kann der Frontend-Thread einen Flaschenhals darstellen.



- Abbildung 5: Teilung in Front- und Backend -

Daher bietet es sich an, auch im Frontend mehrere Threads einzusetzen. Um die Kosten der zusätzlichen Synchronisation zwischen Front- und Backend abzumildern, können bei dieser Variante auch Verarbeitungsschritte mit bekannten Zeitverhalten gleich im Frontend durchgeführt werden. Im Vorliegenden Anwendungsfall wäre dies z. B. die komplette Abhandlung des HTTP-Protokolls.



- Abbildung 6: Verwendung mehrerer Threads -

Fazit:

Mit einer geeignet gewählten Architektur ist das Qt-Framework für die Implementierung von Servern im mittleren Performance-Segment gut geeignet. Die Firma yasc hat sehr gute Erfahrungen in verschiedenen Anwendungen mit Qt-basierten Servern gemacht. U. a. im Einsatz als zentraler Kommunikationsknoten in einem großen Fahrgast-Informationssystem oder als verteiltes Laborsystem für Hardware-In-The-Loop-Tests mit Echtzeitanforderungen. Die reichhaltige und gut strukturierte Qt-Plattform ermöglicht dabei das kosteneffiziente und wartungsarme Umsetzen auch anspruchsvoller Anwendungen.