Hauptseminar: **Peer-to-Peer-Netzwerke und fehlertolerante Algorithmen**

Practical Byzantine Fault Tolerance

Christian Kellermann

Überblick

- Eigenschaften des Algorithmus
- Grundsätzliches
- BFT-PK: Byzantinische Fehlertoleranz mit public-key-Verfahren
- Erweiterungen
- Performance

Eigenschaften d. Algorithmus

- Feste Anzahl teilnehmender Knoten
- Alle Knoten bieten denselben Dienst an (Dienst wird repliziert)
- Tolerant gegenüber byzantinischen Fehlern
- Resistent gegen denial-of-service-Attacken
- exactly-once Semantik

Grundsätzliches

- Unzuverlässiges Netzwerk
- Angebotener Dienst repräsentierbar durch endliche, det. Automaten
- Knoten unterteilt in 1 Primary, sonst Backup Knoten
- Client sendet Anfrage an einen beliebigen Knoten
- Höchstens $\frac{n-1}{3}$ Knoten sind fehlerhaft

Grundsätzliches (2)

- Quoren: bestehen aus $2f + 1$ Knoten
  - Zwei Quoren mind. 1 korrekten Knoten gemeinsam
  - Es gibt immer ein Quorum, das nur aus korrekten K. besteht
- Zertifikate:
  - Quorums-Zert.: Ein Quorum hat best. Nachricht erhalten

Grundsätzliches (2) contd.

- Schwache Zert.: Wenigstens ein korrekter K. hat Nachricht erhalten ($1+f$ Empfangsbestätigungen)

- Primary p in Sicht v: $p = v \bmod |Knotenanzahl|$

BFT-PK: signierte Nachrichten

- Mit public key Verfahren signierte N. (nicht zu fälschen)
- Kollisionsfreie Hash-Funktion erzeugt sog. digests von N.
- Aufgaben primary-Knoten:
  - Annahme von Requests
  - Koordination der Reihenfolge von Requests

BFT-PK: Normale Operation

- Client sendet REQUEST-Nachricht: $\langle REQUEST, o, t, c \rangle_{\delta_c}$
- Übermittlung des Ergebnisses der einzelnen Knoten an Client: $\langle REPLY, v, t, c, i, r \rangle_{\delta_i}$
- Client akzeptiert ein Ergebnis mit schwachem Zert.

BFT-PK: Ordnung der Requests

- Primary schickt $\langle PRE-PREPARE, v, n, m \rangle_{\delta_p}$
- Backups antworten mit $\langle PREPARE, v, n, d, i \rangle_{\delta_i}$
- Bei Quorums-Zertifikat für $PREPARE$: $\langle COMMIT, v, n, d, i \rangle_{\delta_i}$
- Nach Zertifikat f. $COMMIT$: Abarbeitung des Requests und $REPLY$

BFT-PK: Ordnung der Requests (2)

- Nachrichten werden nur akzeptiert, wenn:
  - View, Digests, Requests und Sequenznummer übereinstimmen
  - Sequenznummer innerhalb gewisser Grenzen liegt (water marks)
- Wohlordnung von Requests
- Ausführung des Requests, wenn kein Request mit kleinerer Sequenznummer ansteht.

Logging

- Logs werden in flüchtigem Speicher gehalten
- Jeder Knoten loggt
  - erhaltene Zertifikate
  - versendete Nachrichten
- Wie lange müssen Logs erhalten bleiben?

Garbage Collection

- periodisch, abhängig von der Sequenznummer
- Abgleich mit anderen Knoten per $\langle CHECKPOINT, v, n, d, i \rangle_{\delta_i}$
- schwaches Zertifikat ausreichend für Bestätigung
- Setzen der water marks h und H: $H = h + L$

BFT-PK: View Changes: When primaries turn bad

- Wenn Timeout für Request,
  $\langle VIEW-CHANGE, v+1, n, s, \mathcal{C}, \mathcal{P}, i \rangle_{\delta_i}$
- Nach Zertifikat sendet der neue Primary,
  $\langle NEW-VIEW, v+1, \mathcal{V}, \mathcal{O}, \mathcal{N} \rangle_{\delta_p}$
- Primary wird über View definiert!

BFT-PK: View Changes (2)

- Sequenznummern nach letztem stable checkpoint müssen neu vergeben werden
- Nicht genutzten Sequenznummern wird ein NOP zugewiesen
- Übergang in nächste Sicht.
- Um zu schnelle Sichtwechsel zu vermeiden, wird Timer exponentiell erhöht.

Erweiterungen: Ersetzen der Signaturen

- Public-key-Verfahren sehr langsam, daher Verwenden von symmetrischen Verfahren
- Verwenden von message authentication codes (MACs) jedoch nicht trivial, da weniger mächtig
  - Requests können nicht immer authentifiziert werden
  - Dritte können nicht von der Echtheit einer Nachricht überzeugt werden

Erweiterungen: Ersetzen der Signaturen contd.

- Zertifikate können nicht übergeben werden

Erweiterungen: Ersetzen d. Signaturen (2)

- Authentifizierungs-Vektor
- Explizite Request-Authentifizierung
  - MAC für Knoten ist korrekt
  - Knoten hat bereits f Nachrichten mit korrektem Digest akzeptiert
  - Knoten hat eine gleiche Anfrage von Client erhalten

Erweiterungen: Ersetzen d. Signaturen (2) contd.

- View-Changes werden mit einem Acknowledge bestätigt

Erweiterungen: Proactive Recovery

- Knoten sollen sich bei Fehlverhalten selbst zurücksetzten
- Annahmen:
  - sichere Kryptographie
  - Read-only Speicher für Schlüsselpaare
  - Watchdog Timer
  - Recovery Monitor

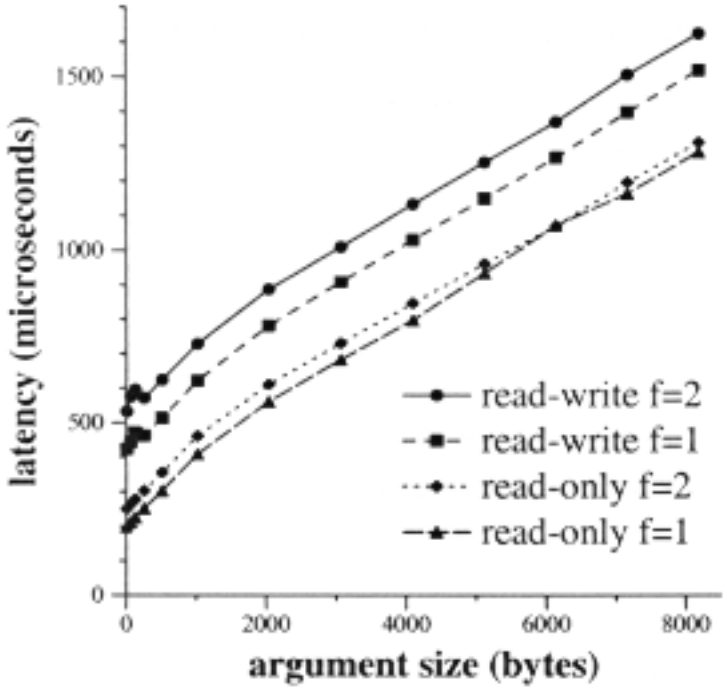
Erweiterungen: Proactive Recovery (2)

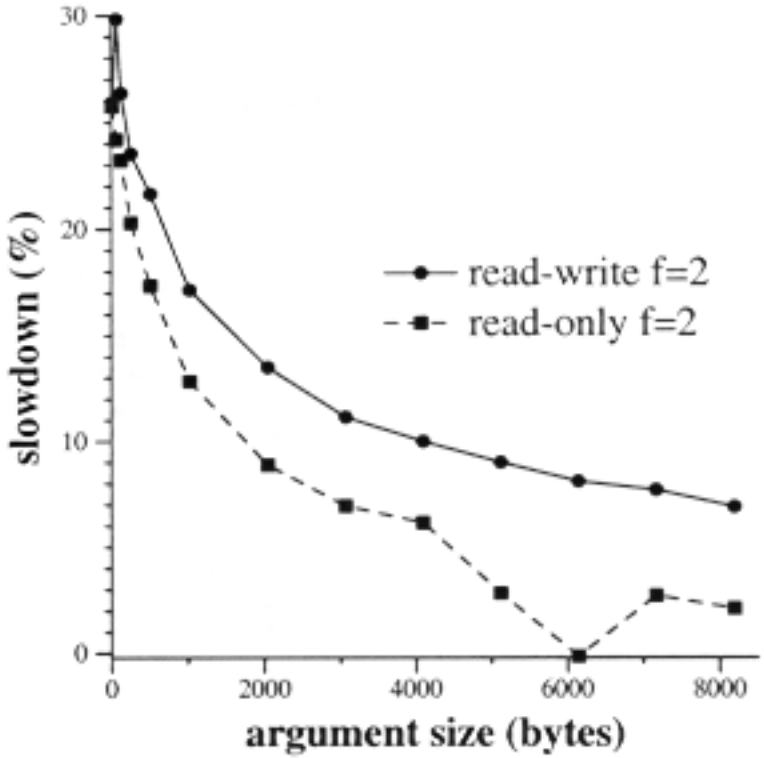
- Periodisches Auffrischen der Session Keys
- Korrektheit des Logs muß überprüft werden
- Bei Recovery muß Log aktualisiert werden

Performance

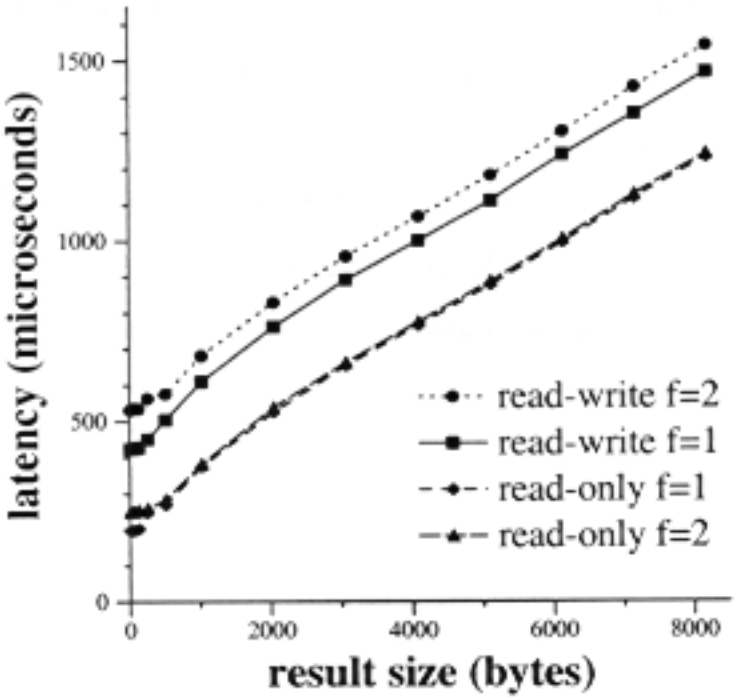
- Testumgebung: 9 Pentium III 600 MHz (2 x 700 MHz), Linux 2.2.16-3 ohne SMP
- Betrachtet wurde u.a. die Verzögerung bei variablen Parameter- und Ergebnisgrößen

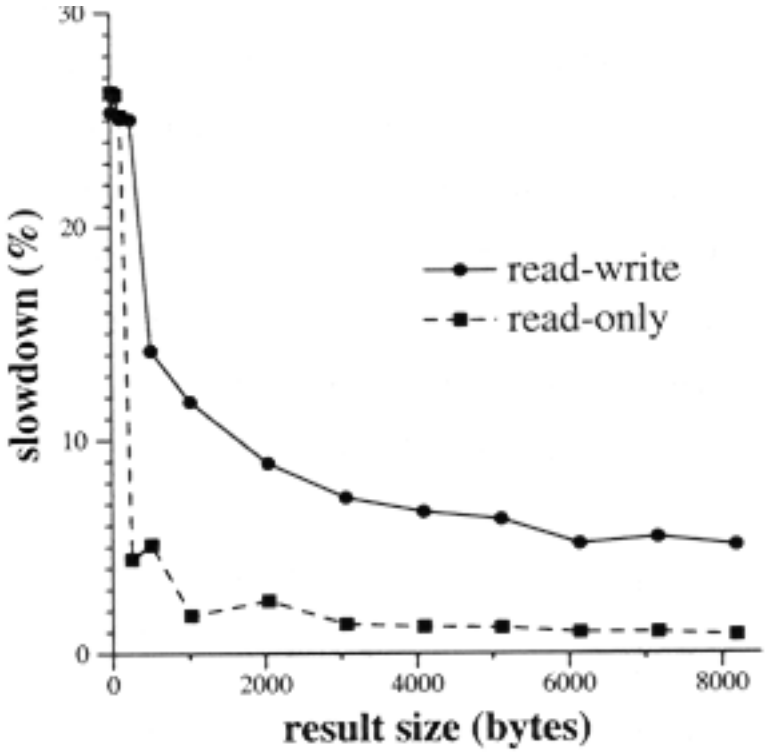
Performance (2)





Performance (3)





Performance (4)

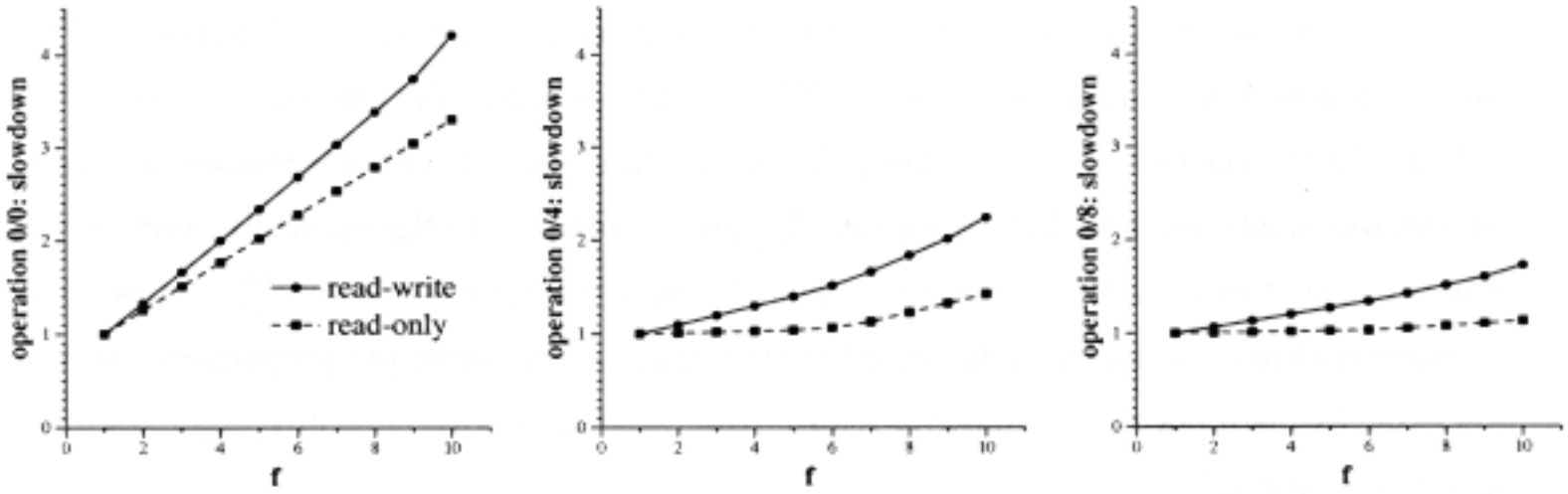


Figure 8-23: Predicted slowdown relative to the configuration with $f = 1$ for increasing f and result size.

Zusammenfassung

- Es dürfen maximal $\frac{n-1}{3}$ Knoten fehlerhaft sein
- Korrektheit und Sicherheit basiert auf Asynchronität
- Um Deadlocks zu vermeiden braucht man Synchronität
- Anzahl der Knoten statisch