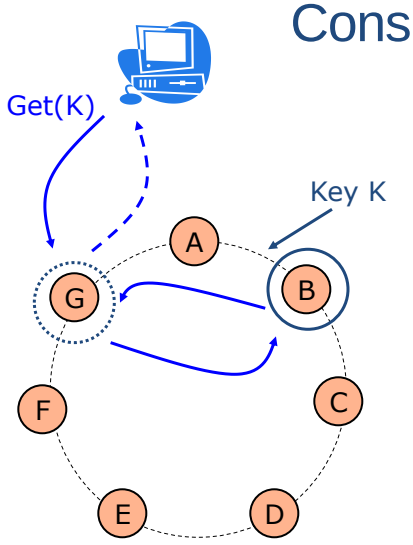
אחסון מבוזר

תזכורת: אתגרים באחסון מבוזר

- Scalability
- ⇦ על נפח האחסון במערכת לגדול בהתאמה לגידול בנתונים
- Load Balancing
- ⇦ התפלגות הגישה לנתונים לא ידועה מראש ועלולים להיווצר צווארי בקבוק
- Decentralization
- ⇦ תלות בשרת מרכזי (למשל לצורך אינדקס) עלולה לעצור את המערכת במקרה של נפילה, עומס זמני, או גידול
- Failure Handling
- ⇦ **תמיד** יש נפילות: חומרה, תוכנה, תקשורת ועומס

תזכורת: Consistent Hashing



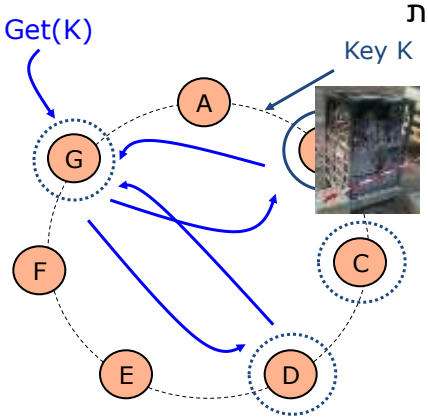
- אובייקט עם מפתח K נמצא בשרת הראשון שהמפתח שלו גדול מ-K
  - ✓קל למצוא אובייקט
  - ✓העומס מחולק בצורה אחידה
- בקשת גישה לאובייקט מנותבת לשרת המחזיק באובייקט
- בהוספת שרת מגרילים עבורו מפתח ומודיעים לשאר השרתים
- בהוצאת שרת מעבירים את האובייקטים שלו לשכן העוקב

בחזרה לאתגרים

- Scalability
- ⇦ קל להוסיף שרתים חדשים: הוספת שרת משפיעה רק על שכן לוגי אחד
- Load Balancing
- ⇦ האובייקטים מפולגים אקראית בין שרתים ניתן לנתב בקשות לשרתים שונים
- Decentralization
- ⇦ אין אינדקס מרכזי, כל שרת יודע למצוא כל אובייקט ב-hop אחד
- Failure Handling
- ⇦אם שרת נופל כל האובייקטים שלו לא נגישים/אובדים!
- ⇦הפתרון: שכפול

שכפול (Replication)

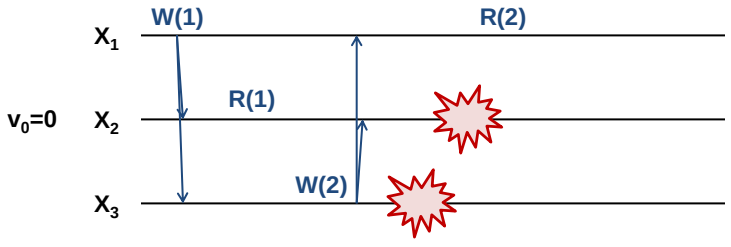
- לכל אובייקט נשמור N עותקים (replicas) על N שרתים (פיזיים) שונים
- אין איבוד מידע בנפילת שרת סופית
- אפשר לגשת למידע גם כששרת נופל זמנית



- איך דואגים שכל העותקים יישארו זהים?

שכפול: המצב הרצוי

- כל כתיבה לאחד העותקים מעדכנת **מיידית** את שאר העותקים
- ⇦ קריאה מכל העותקים מחזירה ערך זהה
- ⇦ אין איבוד מידע בנפילת N-1 שרתים או פחות
- ⇦ אפשר לגשת למידע גם בנפילה זמנית של N-1 שרתים או פחות



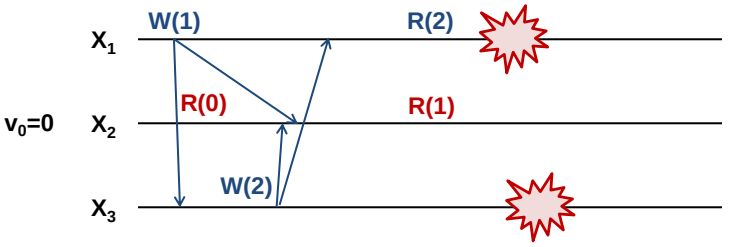
שכפול: המצב המצוי

- לא ניתן להזניח את זמן העדכון
- עדכונים עלולים להגיע שלא על פי סדר ביצוע הכתיבה

⇦עותקים שונים עלולים להיות מעודכנים "במקביל" בערכים שונים

⇦איזה ערך לשמור?

- הנתונים לא תמיד עקביים



עקביות עם ריבוי עותקים

- עקביות עבור אובייקטים שונה מעקביות עבור תנועות
- נתייחס לתכונה הבאה:

אם פעולת קריאה החלה לאחר שפעולת כתיבה **הסתיימה**, הערך שיוחזר הוא הערך שנכתב בכתיבה זו, או בפעולת כתיבה **מאוחרת** יותר

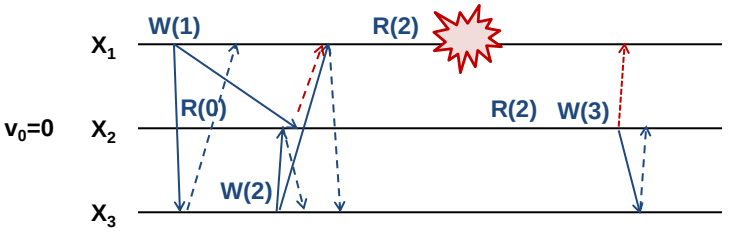
- זוהי תכונה הכרחית אבל לא מספיקה
- קיימות הגדרות שונות לעקביות שלא נתעמק בהן כאן

שמירה על עקביות (גישה 1)

- נחכה לאישור על הכתיבה מכל השרתים שמחזיקים עותקים
- קריאה מכל אחד מהשרתים תבטיח החזרת ערך נכון
- ניתן להשוות חותמות זמן על מנת לפתור קונפליקטים

⇦קריאה מהירה (תמיד ניתן לקרוא)

⇦כתיבה אטית... (לא תמיד ניתן לכתוב)

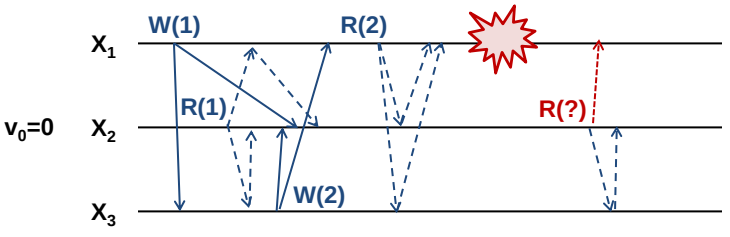


שמירה על עקביות (גישה 2)

- כתיבה יכולה להתבצע גם על שרת בודד (לא מחכים לאישור)
- קריאה צריכה להתבצע מכל השרתים כדי להבטיח ערך נכון

⇦כתיבה מהירה (תמיד ניתן לכתוב)

⇦קריאה אטית... (לא תמיד ניתן לקרוא)



שמירה על עקביות (מקרה כללי)

- האובייקט משוכפל על N שרתים
- קריאה מחייבת גישה ל-R עותקים
- כתיבה מחייבת גישה ל-W עותקים

W+R>N

⇦לפחות אחת משתי הפעולות מתבצעת על **רוב** השרתים

✓בקריאה תמיד ניגש לשרת שראה את הכתיבה האחרונה שהסתיימה לפני שהתחלנו

- N גדול מגדיל את השרידות
- בשילוב עם W גדול מגדיל את הזמינות לקריאה (אבל הכתיבה אטית או לא זמינה)
- בשילוב עם W קטן מגדיל את הזמינות לכתיבה (אבל הקריאה אטית או לא זמינה)

זמינות (Availablity)

- כל פעולה שמגיעה לשרת תקין חייבת להתבצע
- Network partition (חלוקה): קיים נתק בקו תקשורת כך שבאופן מעשי השרתים מחולקים לשתי קבוצות ללא קשר ביניהן

⇦בקבוצה המכילה את רוב השרתים ניתן לבצע לפחות קריאה **או** כתיבה

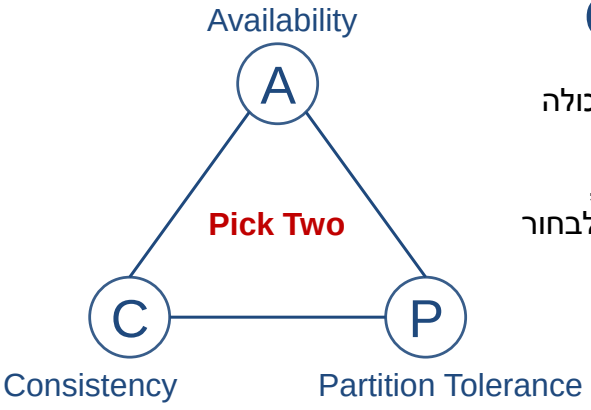
⇦בקבוצה השניה ניתן (אולי) לבצע רק אחת מן הפעולות

⇦תמיד יהיו שרתים שלא יכולים לבצע פעולות

- במערכות עסקיות מסויימות, זמינות היא התכונה החשובה ביותר!

"...customers should be able to **view and add items to their shopping cart** even if disks are failing, network routes are flapping, or data centers are being destroyed by tornadoes."
(Amazon)

CAP Theorem



- מערכת זמינה ועקבית לא יכולה לעמוד בפני נפילות רשת
- מכיוון שנפילות רשת קורות, במערכות אחסון מבוזר יש לבחור בין זמינות לעקביות
- אפשר לוותר על עקביות?

~~תכונות ACID~~ זמינות

- אטומיות
  - ~~כל הפעולות של תנועה מתבצעות/לא מתבצעות במקשה אחת~~
  - פעולות על אובייקטים בודדים
- עקביות
  - ~~ביצוע התנועה (או ביטולה) שומר על נכונות מסד הנתונים~~
  - מוחלשת לטובת זמינות
- בידוד
  - ~~לא רואים/ניגשים לנתונים במצב ביניים (לא עקבי)~~
  - לא מובטח
- שרידות
  - אפקט של תנועה שהסתיימה בהצלחה לא נעלם

Eventual Consistency

עקביות בסופו של דבר

- בסופו של דבר כל העותקים מקבלים את כל העדכונים
- אם אין עדכונים חדשים, בסופו של דבר כל העותקים יתכנסו לאותו ערך

דוגמאות

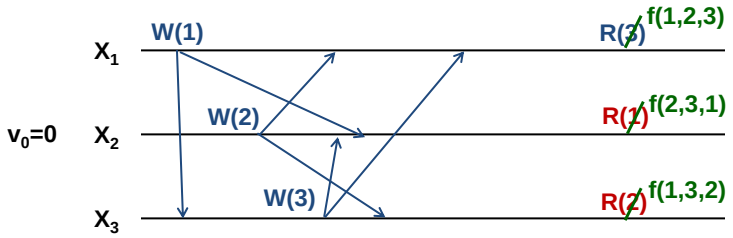
- Domain Name System (DNS)
  - שרתי מטמון שומרים כתובות נפוצות ומעודכנים בתדירות קבועה
- מכירה של מוצר שלא קיים במלאי
  - העסקה מתבטלת, ניתן החזר ואולי פיצוי
- חשבונות בנק
  - המחאות חוזרות (עם קנס)
  - משיכה בכספומט מוגבלת בסכום

קל להחליט מה הערך הנכון

Eventual Consistency

עקביות בסופו של דבר

- בסופו של דבר כל העותקים מקבלים את כל העדכונים
- אם אין עדכונים חדשים, בסופו של דבר כל העותקים יתכנסו לאותו ערך



⇦איך נטפל בעדכונים סותרים?

יישוב סתירות (Conflict Resolution)

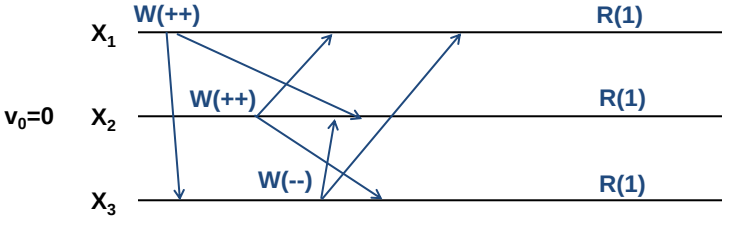
- ההסטוריה של עותק היא קבוצת כל העדכונים שהעותק קיבל
- עשויה להכיל עדכונים סותרים: יש להכריע ביניהם בלי תלות בסדר הגעתם

שיטות ליישוב סתירות

- מבני נתונים "קומוטטיביים"
  - כל הפעולות קומוטטיביות ולכן על פי הגדרה אין חשיבות לסדר ביצוען
  - מונים, קבוצות, וגרפים עם פעולות מסויימות
- העברת ההחלטה לאפליקציה
  - בקריאה מחזירים הסטוריה במקום ערך, והאפליקציה משתמשת בידע נוסף כדי ליישב סתירות
- שעונים וקטוריים (vector clocks)
  - כל עדכון מכיל סיכום של ההסטוריה שהעדכון הסתמך עליה
  - לא תמיד ניתן להכריע בעזרתם
  - עוד בקורס "תכנות מקבילי ומבוזר"

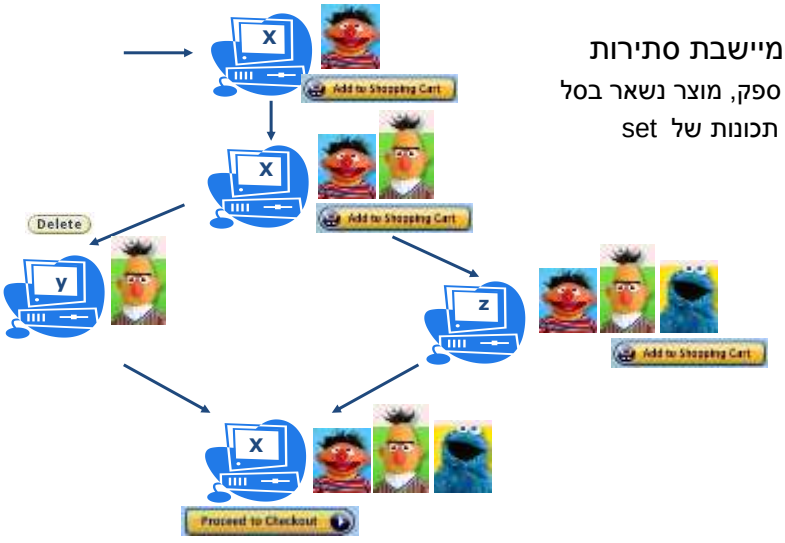
דוגמא: מונה

- מבנה נתונים קומוטטיבי
  - Increment (++)
  - Decrement (--)
- בתנאי שאין גלישה (overflow)...
- מונה כללי: חשבון בנק



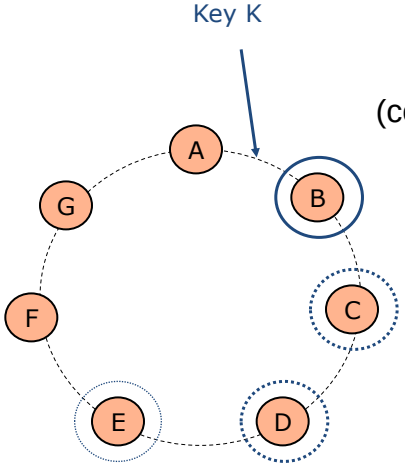
דוגמא: Shopping Cart

- האפליקציה מיישבת סתירות
  - במקרה של ספק, מוצר נשאר בסל
  - לא נשמרות תכונות של set

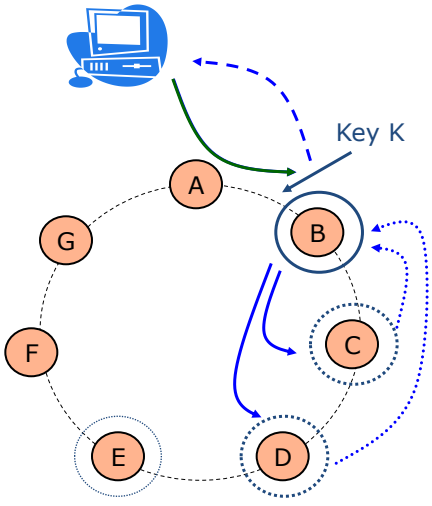


ערבול עם שכפול

- אובייקט משוכפל על N שרתים
  - N נבחר על פי האפליקציה
  - N שרתים פיזיים
- השרת הראשון הוא המתאם (coordinator)
- שאר השרתים הם רשימת העדיפות
  - הרשימה ידועה לכל השרתים בטבעת
  - הרשימה כוללת יותר מ-N שרתים כדי להתמודד עם נפילות

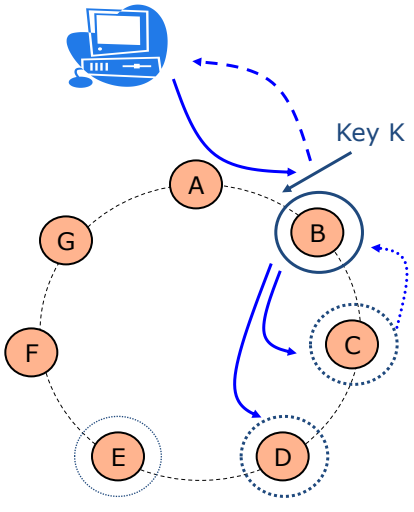


ביצוע Get()



- המתאם קורא גרסאות מ-N השרתים הזמינים הראשונים
- מחכה ל- R-1 תגובות
- מצמצם גרסאות שאין ביניהן קונפליקטים
- מחזיר לאפליקציה גרסאות סותרות
- האפליקציה מחזירה ערך מיושב

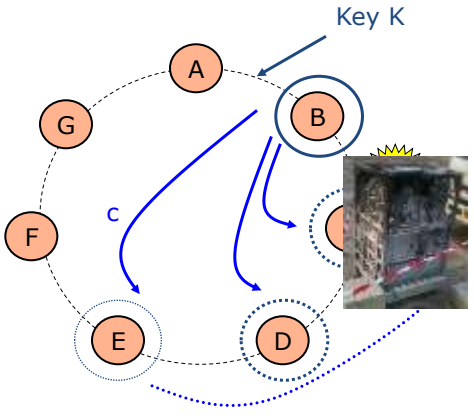
ביצוע Put()



- המתאם מייצר גרסא חדשה
- הגרסא נשלחת ל-N השרתים הזמינים הראשונים
- מחכה ל- W-1 תגובות

ביצוע פעולות בעת נפילה זמנית

- הפעולה מבוצעת על N השרתים הזמינים הראשונים
  - כזכור, ברשימת העדיפויות יש יותר מ-N
- עותק ששייך לשרת שנפל מסומן ככזה ומאוחסן בנפרד
- כאשר השרת חוזר לפעילות העותקים נשלחים אליו (באיחור)
- צריך ליישב גרסאות סותרות



סיכום: זמינות כפונקציית מטרה

- "Always writable"
  - כתיבות יכולות להתבצע גם אם חלק מהעותקים לא זמין
  - פעולת Put() לעולם לא אובדת
  - הסיבוכיות הנובעת מסתירות עוברת לקריאה במקום לכתיבה
- גישה אופטימית
  - ברב המכריע של המקרים אין סתירות
  - קיים פרוטוקול מוגדר היטב ליישוב סתירות
    - המערכת פותרת סתירות ברמה המבנית
    - האפליקציה פותרת סתירות ברמה הסמנטית

Percent of requests	Number of versions
99.94	1
0.00057	2
0.00047	3
0.00009	4

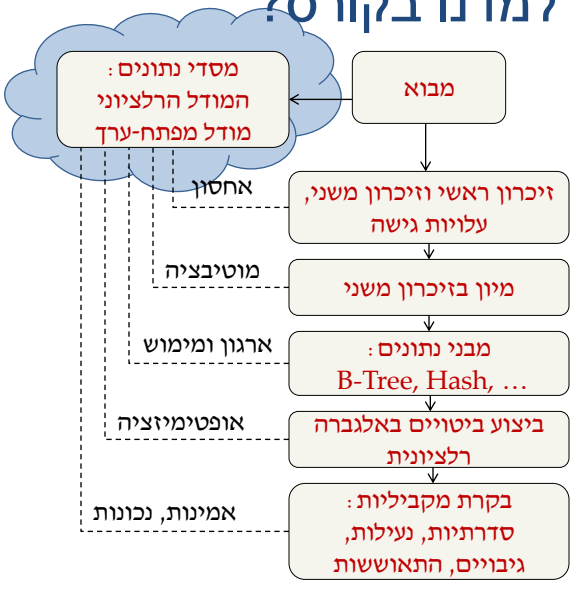
Shopping Cart experiment (24 hours)

גישות נוספות לאחסון מבוזר

הגישה של Amazon ב-Dynamo היא אחת מני רבות, למשל

- Network File System (NFS)
  - פרוטוקול פתוח למערכת קבצים מבוססת client-server
  - שכפול נכנס רק בגרסא מאוחרת
- Google File System (GFS)
  - ארכיטקטורת master-slave בהנחה שכתיבות הן מאורע נדיר (לשימוש פנימי בלבד)
- Hadoop Distributed File System (HDFS)
  - קבצים מחולקים בין שרתים אך גישה אליהם עוברת דרך name node מרכזי
  - מופץ כקוד פתוח בליווי מימוש של MapReduce
- OceanStore
  - Key Value Store הממומש על שרתים עצמאיים בפרוטוקול peer-to-peer
  - עותקים של האובייקט נשמרים היכן שיש גישות רבות, ומאותרים בפרוטוקול ניתוב
  - מגן מפני נפילות והתקפות על ידי שמירת גרסאות ומנגנונים קריפטוגרפיים

סיכום: מה למדנו בקורס?



מה זה ענן? המודל

- מערכת אחסון מבוזר עם פרישה רחבה
- כל החומרה, וחלקים מן התוכנה שייכים לספק (cloud provider)
- המשתמשים שוכרים שרתים **וירטואליים** (יחידות חישוב, אחסון ותקשורת) בצורה של חבילת שירות
  - תשלום על פי המשאבים המשוריינים למשתמש
  - תשלום על פי תעבורת רשת בפועל

- במקור: Infrastructure as a Service (IaaS Cloud)
- לרב: Platform as a Service (PaaS)
- יותר ויותר: Software as a Service (SaaS)
  - למשל Data Base as a Service (DBaaS)
  - Gmail, Dropbox, etc.

דוגמא: Amazon Elastic Compute Cloud (EC2)

On-Demand Instance Prices

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
m3.xlarge	4	13	15	2 x 40 SSD	$0.450 per Hour
m3.2xlarge	8	26	30	2 x 80 SSD	$0.900 per Hour
Compute Optimized - Current Generation					
c3.large	2	7	3.75	2 x 16 SSD	$0.150 per Hour
c3.xlarge	4	14	7.5	2 x 40 SSD	$0.300 per Hour
c3.2xlarge	8	28	15	2 x 80 SSD	$0.600 per Hour
c3.4xlarge	16	55	30	2 x 160 SSD	$1.200 per Hour
c3.8xlarge	32	108	60	2 x 320 SSD	$2.400 per Hour
Memory Optimized - Current Generation					
m2.xlarge	2	6.5	17.1	1 x 420	$0.410 per Hour
m2.2xlarge	4	13	34.2	1 x 850	$0.820 per Hour
m2.4xlarge	8	26	68.4	2 x 840	$1.640 per Hour
Storage Optimized - Current Generation					
i2.xlarge	4	14	30.5	1 x 800 SSD	$0.853 per Hour
i2.2xlarge	8	27	61	2 x 800 SSD	$1.705 per Hour
i2.4xlarge	16	53	122	4 x 800 SSD	$3.410 per Hour
i2.8xlarge	32	104	244	8 x 800 SSD	$6.820 per Hour

נתונים חלקיים, נכון לינואר 2014

דוגמא: Amazon Elastic Compute Cloud (EC2)

Heavy Utilization Reserved Instances

	1 yr Term		3 yr Term	
	Upfront	Hourly	Upfront	Hourly
General Purpose - Current Generation				
m3.xlarge	$1266	$0.105 per Hour	$1922	$0.086 per Hour
m3.2xlarge	$2531	$0.209 per Hour	$3844	$0.172 per Hour
Compute Optimized - Current Generation				
c3.large	$466	$0.037 per Hour	$726	$0.032 per Hour
c3.xlarge	$932	$0.075 per Hour	$1451	$0.064 per Hour
c3.2xlarge	$1863	$0.149 per Hour	$2902	$0.128 per Hour
c3.4xlarge	$3726	$0.298 per Hour	$5804	$0.257 per Hour
c3.8xlarge	$7452	$0.596 per Hour	$11609	$0.513 per Hour
Memory Optimized - Current Generation				
m2.xlarge	$789	$0.068 per Hour	$1198	$0.053 per Hour
m2.2xlarge	$1578	$0.136 per Hour	$2396	$0.106 per Hour
m2.4xlarge	$3156	$0.272 per Hour	$4792	$0.212 per Hour
Storage Optimized - Current Generation				
i2.xlarge	$1820	$0.155 per Hour	$2740	$0.121 per Hour
i2.2xlarge	$3640	$0.311 per Hour	$5480	$0.241 per Hour
i2.4xlarge	$7280	$0.621 per Hour	$10960	$0.482 per Hour
i2.8xlarge	$14560	$1.242 per Hour	$21920	$0.964 per Hour

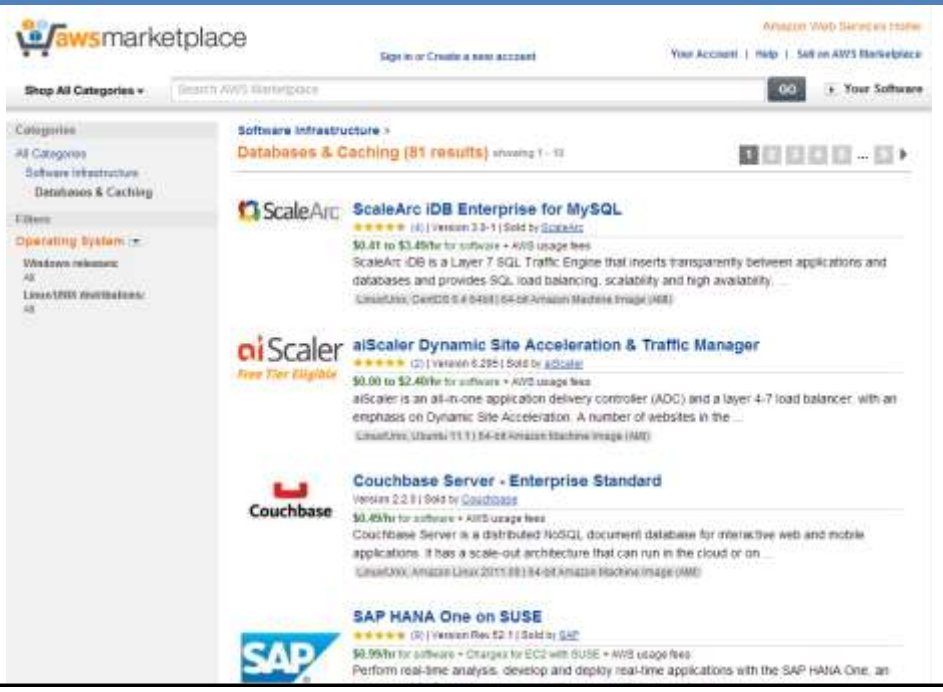
נתונים חלקיים, נכון לינואר 2014

דוגמא: Amazon Elastic Compute Cloud (EC2)

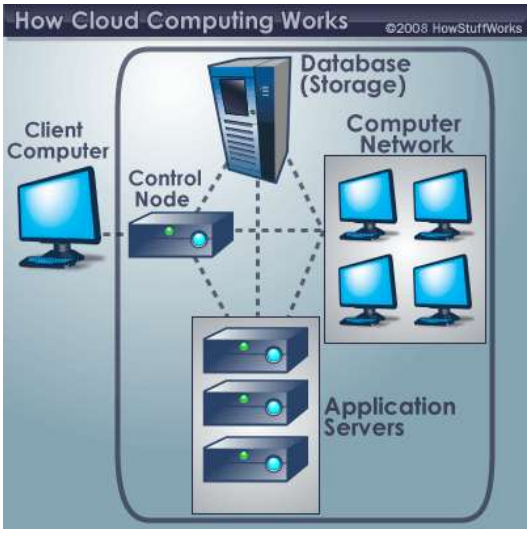
Data Transfer Rates

Data Transfer IN To Amazon EC2 From	
Internet	$0.00 per GB
Another AWS Region (from any AWS Service)	$0.00 per GB
Amazon S3, Amazon Glacier, Amazon DynamoDB, Amazon SQS, or Amazon SimpleDB in the same AWS Region	$0.00 per GB
Amazon EC2, Amazon RDS and Amazon ElastiCache instances or Elastic Network Interfaces Using a public or Elastic IP address	$0.01 per GB
Data Transfer OUT From Amazon EC2 To	
Amazon S3, Amazon Glacier, Amazon DynamoDB, Amazon SQS, Amazon SimpleDB in the same AWS Region	$0.00 per GB
Amazon EC2, Amazon RDS, or Amazon ElastiCache instances, Amazon Elastic Load Balancing, or Elastic Network Using a public or Elastic IP address	$0.01 per GB
Another AWS Region or Amazon CloudFront	$0.02 per GB
Data Transfer OUT From Amazon EC2 To Internet	
First 1 GB / month	$0.00 per GB
Up to 10 TB / month	$0.12 per GB
Next 40 TB / month	$0.09 per GB
Next 100 TB / month	$0.07 per GB
Next 350 TB / month	$0.05 per GB

נתונים חלקיים, נכון לינואר 2014



מה זה ענן? המימוש



- מערכת האחסון ושרתי החישוב הם ה-backend
- שרתים מיוחדים מנהלים את המשאבים ואת גישות המשתמשים

היתרון הכלכלי:

- איחוד (consolidation)
  - יתרונות של סיטונאות
  - בניהול נכון ניתן לחסוך משאבים
- תקשורת זולה
  - הענן "צמוד" לאינטרנט
  - ספק הענן הוא ספק התקשורת

הענן דומה למערכות אחסון שהכרנו בקורס

- למרות השימוש הנרחב ב-SSD, האחסון הקבוע הוא על דיסקים
  - נעשה שימוש נרחב באופטימיזציות של זיכרון מטמון
  - **אבל** למשתמש הקצה אין מושג איך הנתונים מאוחסנים ופרוסים
- ספקי השירות אחראים על שרידות ועמידות בנפילות
  - נעשה שימוש נרחב בקונפיגורציות RAID
  - **אבל** נפוץ מאד שכפול באתרים מרוחקים (off site) על כל המשתמע מכך
- מסדי הנתונים צריכים לתמוך במקביליות גבוהה של תנועות
  - רבים תומכים במודל הרלציוני ובתכונות ה-ACID שהוא דורש (בוריאציות שונות)
  - **אבל** יותר ויותר גרסאות מבוזרות מספקות עקביות מוחלשת לטובת זמינות

סיכום: הענן מציב אתגרים חדשים

- למשתמש: אופטימיזציות של ביצועים מורכבות יותר
  - בחירה בין מערכת אחסון מקומית לבין שימוש בענן
  - בחירת הספק וחבילת השירות המתאימים
  - כמו בעבר, בחירה טובה תלויה בהערכת הדרישות והביצועים של המערכת
- הספק צריך לעמוד בתנאי השירות המובטחים בחבילה במינימום עלות
  - Service Level Agreement (SLA)
  - בידוד בהיבט של ביצועים (Performance Isolation)
- תקשורת משחקת תפקיד מרכזי מאד
  - עמידות בפני נפילות רשת
  - אופטימיזציות ניתוב על סמך עומסים קבועים או זמניים ועלויות של מקטעים שונים
- אבטחת מידע
  - דיברנו על עקביות ועמידות בפני נפילות
  - בענן (ובכל מערכת מבוזרת) קיימים איומים נוספים: וירוסים, פריצות וכו'
  - שמירה על פרטיות

הבחינה

- יום שישי 31.1.2014

- 3 שאלות
- חומר פתוח
- כל מה שנלמד בהרצאות ובתרגולים

- שעות קבלה יפורסמו באתר הקורס