Cloud Computing e dispositivi mobili

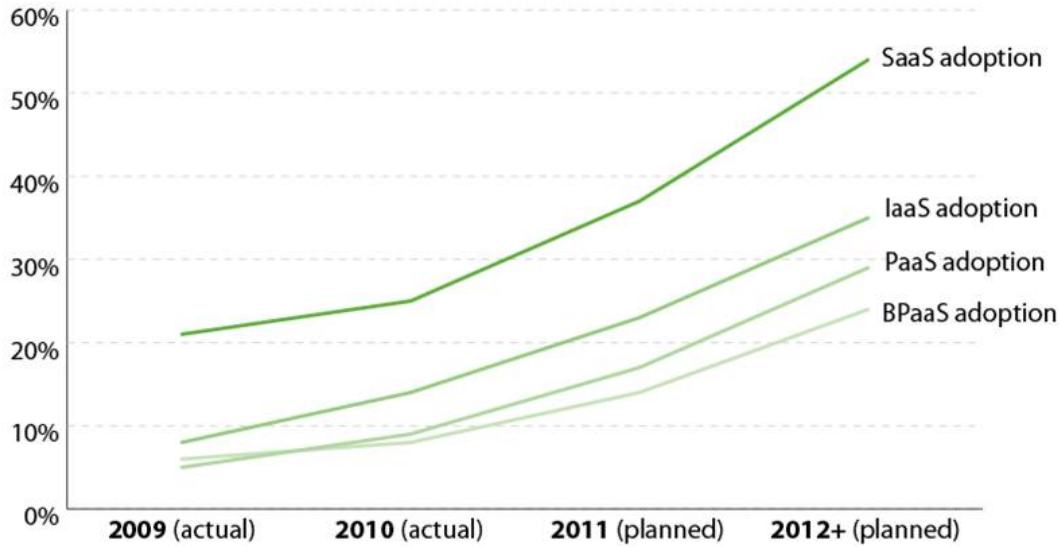
Seminario per il corso di Sistemi Middleware

Marco Di Nicola
marco.dinicola@studio.unibo.it

Sviluppo del Cloud Computing

- Una notevole migrazione di applicativi/dati su architetture Cloud negli ultimi anni.
  - ✓ Costi ridotti, maggiore flessibilità e stabilità.

Cloud computing adoption rates and future plans



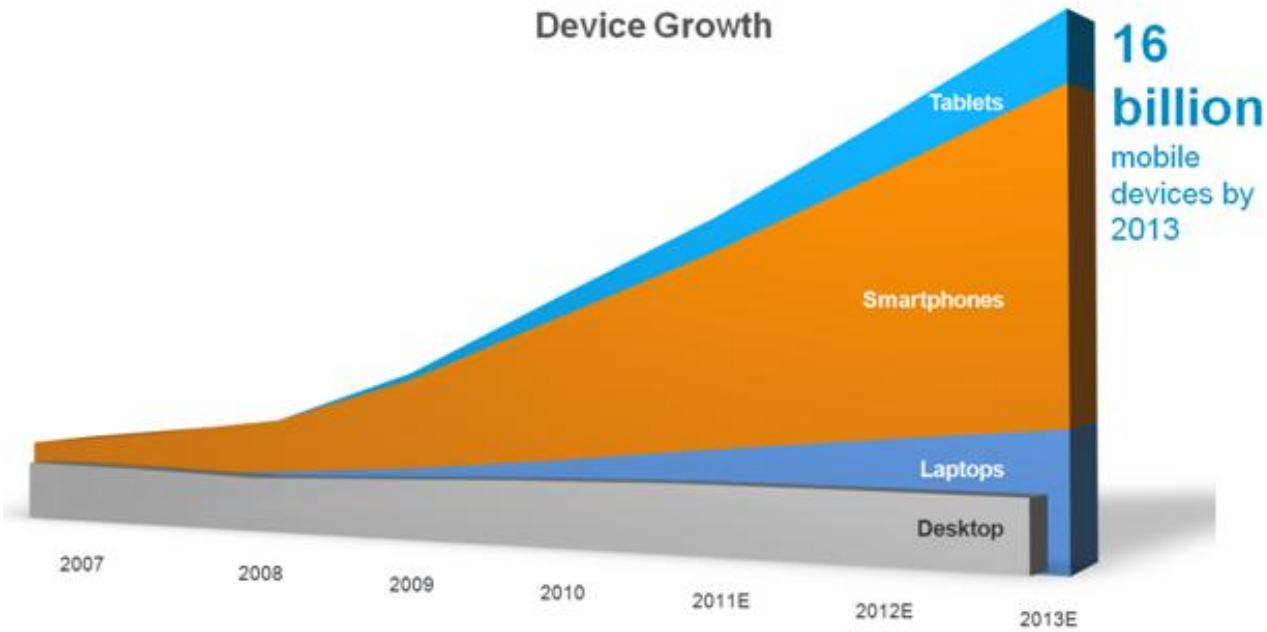
	2009 (actual)	2010 (actual)	2011 (planned)	2012+ (planned)
SaaS adoption	21%	25%	37%	54%
IaaS adoption	8%	14%	23%	35%
PaaS adoption	5%	9%	17%	29%
BPaaS adoption	6%	8%	14%	24%

Base: 531 North American and European software decision-makers

Source: Enterprise And SMB Software Survey, North America And Europe, Q4 2009; Forrsights Software Survey, Q4 2010

Diffusione di dispositivi mobili

- Sostanziale incremento nell'utilizzo di dispositivi mobili (smartphones, tablets, laptops, . . .) negli ultimi anni.
  - √ Portabilità ed accesso costante a dati ed informazioni.



Source: Gartner Research; Smartphone, Tablet, and PC Forecast, December 2010

Mobile e Cloud Computing

- Ad oggi è registrato un sempre maggiore numero di dispositivi mobili che utilizzano servizi basati su Cloud Computing, rispetto a PC Desktop.
- Questioni di interesse:
  - Potenzialità date dall'interazione tra i due paradigmi e relative necessità progettuali.
    Quindi le tecnologie che appaiono più promettenti a questo scopo.
  - Stabilità e migliori performance attraverso l'uso di architetture specializzate.
  - Gestione di problematiche derivanti dalla mobilità dei dispositivi.

Potenzialità dei dispositivi

- **Smart Mobile Devices** (SMD)
  - Capacità di instaurare connessioni TCP in ogni luogo, utilizzando eventualmente diverse interfacce di rete (802.11x, 3G, ...).
  - Possibilità di inferire il contesto attuale di utilizzo attraverso l'uso di sensori di varia natura.
  - Disponibilità di un ambiente di sviluppo facile da utilizzare e basato su linguaggi di programmazione moderni (es: J2ME, Cocoa Touch). Approccio di alto livello a connessione TCP/IP, parsing di documenti XML ed un framework per la creazione di interfacce utente.

Context Aware applications

- Applicazioni capaci di operare in maniera del tutto autonoma ed indipendente dall'utilizzatore, dotate di un alto grado di adattabilità.
- Distinzione di vari contesti di utilizzo tramite i sensori del dispositivo.
  - Utilizzo di differenti applicazioni in base al contesto attuale (es: business e privato).
  - Rilevazione autonoma del contesto attuale in base alle periferiche esterne rilevate o dati quali temperatura, illuminamento, etc .

Contesti di interesse

- Contesti spaziali.
  - Basati su posizione, prossimità e direzione di spostamento.
  - Inferiti a partire da GPS, accelerometro, compasso, etc.
- Contesti di attività.
  - Descrivono l'attività (o più di una) che l'utente sta eseguendo in un determinato momento: correre, guidare, ascoltare musica, etc.
  - Inferiti a partire da molteplici sensori sul dispositivo, applicazioni in esecuzione e azioni esplicite dell'utente.
- Contesti sociali.
  - Composizioni di più contesti di attività di diversi individui, rappresentano interazioni tra utenti.
  - Informazioni collezionate in modo implicito (applicazioni Cloud) o esplicito (web-services messi a disposizione da diversi social networks).

Utilizzi del Cloud: Storage

- **Cloud storage**: spostare su Cloud i dati non necessari nell'immediato.
- Necessità di ottimizzare la dimensione dei dati da trasferire.
  Variabili da considerare:
  - Tipo di connessione dati e banda disponibile.
  - Memoria disponibile (determina la dimensione dei pacchetti di dati trasferiti).
  - Dimensione logica di un blocco di informazioni.
- Persistenza di dati memorizzati per storage, quindi per eventuale uso futuro.
- Disponibilità di dati necessari ad un'applicazione in esecuzione, per il completamento di un determinato task.
  Fattori che introducono latenza:
  - Tempo di trasferimento.
  - Negoziazione della connessione.
  - Encoding e decoding binario.

Utilizzi del Cloud: Processing

- **Cloud processing**: esecuzione di applicazioni su piattaforme Cloud, riducendo la computazione su dispositivi.
  - √ Adatto a tasks che utilizzano intensivamente il processore.
- Utilizzo di un meccanismo asincrono di richieste e push/callbacks, al fine di evitare attese da parte del dispositivo.
- Minimizzare i tempi di attesa dell'utente.
  - Tener conto dell'overhead introdotto da comunicazione con Cloud.
  - Spostare la computazione di tasks sufficientemente complessi.

Utilizzi del Cloud: Security

- **Cloud security**: spostare dati sensibili su dispositivi più facili da proteggere ed isolare di un dispositivo smart, quali Laptops e Desktops.

- Occorrono meccanismi di autenticazione con il provider Cloud:
  - OpenID e OpenAuth.
    Usate insieme e particolarmente sicure in quanto rimuovono la necessità di conservare le credenziali sul dispositivo mobile.
  - User request token.
    Usata da Google e Yahoo geocoding, risulta nel semplice passaggio di un token di autenticazione nell'URI di una richiesta.
  - Metodi ad hoc che comprendono la firma di porzioni degli headers delle richieste, usando diversi algoritmi crittografici (es: Amazon AWS).

RESTful Web Services

- **RE**presentative **S**tate **T**ransfer: architettura software basata su semplice protocollo Client-Server di richiesta-risposta.
  - √ Stateless: minimizza problemi dovuti alla mobilità dei dispositivi (volatilità della rete).
  - √ Utilizza risposte HTTP: come sopra, grazie all'uso di messaggi discreti.
  - √ Basato su URL: facile da invocare.
  - √ Prevede messaggi concisi: riduce overhead da protocollo e quindi si presta ad ambienti con scarsa memoria.
- Il payload delle risposte può contenere semplici dati binari, testo, documenti HTML o XML.

Processing di XML

- La struttura di un documento XML consente di comunicare informazioni complesse in un formato semi-strutturato.
- Costituiscono la maggior parte delle risposte REST, in sistemi discretamente complessi.
- Due metodi fondamentali per il parsing:
  - Utilizzo di un Document Object Model: caricamento in memoria di una struttura ad albero che rappresenta l'intero documento.
  - Modello Event-Driven che utilizza un insieme di callbacks relative agli elementi sintattici incontrati nel parsing del documento: tags iniziali, tags finali, testo, commenti, etc.
    - √ Adatto a dispositivi smart con poca memoria, in quanto permette di ignorare dati non necessari mentre il parsing è in esecuzione.

REST: Considerazioni

- Occorre determinare il rapporto ottimale tra la frequenza delle richieste e loro dimensione, quindi quantità di dati da inviare nel payload di un singolo messaggio.
  Fattori da considerare:
  - Volatilità della rete nei dispositivi smart.
  - Overhead da protocollo e da parsing di XML: ridurre tempi di attesa dell'utente.

- Bisogna considerare che esistenza della rete e velocità di connessione non sono garantite nei dispositivi smart.
  - Invio di singoli pacchetti al Server per comunicare presenza sulla rete.
  - Download di piccoli oggetti per monitorare la latenza di ricezione.

Aspettative di utilizzo di dispositivi mobili

- **Social Computing**: da interazioni "Human-Computer" a "Human-Service".
  - Possibilità di svolgere diverse attività in modo indipendente dal dispositivo in utilizzo.
    (Leggere le news, acquistare biglietti ferroviari, verificare la propria posizione geografica, . . .)
  - Accesso costante (tramite rete) a dati ed informazioni.

- Possibilità concretizzata mediante utilizzo di web applications e servizi basati sul paradigma del cloud computing.

Mobilità vs Usabilità

- Riducendo la dimensione dei dispositivi la mobilità migliora, ma l'usabilità ne risente, a causa di:
  - Piccole dimensioni dei display
  - Interfacce di I/O limitate.

- Soluzione: connettere periferiche esterne quali schermi e tastiere al dispositivo, attraverso protocolli wireless, onde migliorare l'esperienza utente.

Interazione con periferiche esterne

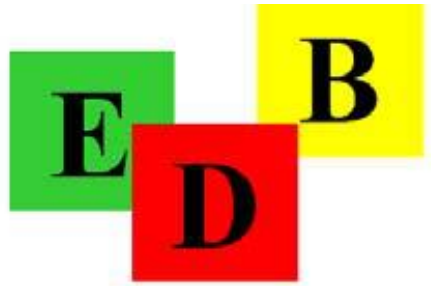
- Necessità di paradigmi per l'accoppiamento wireless di dispositivi semplici e sicuri.
  - Near Field Communication (prossimità).
  - Touch.
- Controllo sulle periferiche esterne.
  - Keyboard e mouse esterni.
  - Sensori presenti sul dispositivo, quali accelerometro e display capacitativo.

Air Mouse Pro trasforma un iphone in controller wireless.

Focus

- **Focus** [1]: applicazione context-aware per Social Computing.
- Distingue diverse categorie di attività dell'utente: comunicazione, informazione, navigazione web e multimedia.
- Definisce diversi contesti di utilizzo: privato, semi-privato, pubblico e business.

Private context



Business context

- Diverse applicazioni in fore/background, con diversi contenuti, a seconda del contesto attuale.
  - Contesto inferito automaticamente in base alle periferiche rilevate (es: schermo d'ufficio o domestico) o selezionato dall'utente.

Cloud come potenziamento di dispositivi

- Considerazioni:
  - I dispositivi mobili continuano a soffrire di varie limitazioni: CPU a bassa frequenza, memoria ridotta e scarsa autonomia garantita.
  - × Non consentono applicazioni che eseguono calcoli intensivi e/o utilizzano molta memoria (rendering offline, gestione di grandi quantità di dati).
  - Cloud computing visto come vasta e scalabile piattaforma, attraverso la quale i providers mettono a disposizione dei consumatori vari servizi, quali piattaforme (*PaaS*), infrastrutture (*IaaS*) e singoli software (*SaaS*).
- Nuova prospettiva: Cloud Computing come mezzo per estendere le capacità di dispositivi (mobili) dalle risorse limitate.
- Diversi possibili approcci.

Una prima possibilità

- Duplicare l'ambiente di esecuzione del dispositivo in una virtual machine, quindi isolata e più sicura, sulla piattaforma Cloud: una **cloudlet**.

  - √ Un aumento nelle risorse di CPU e memoria.
  - √ Approccio trasparente alle applicazioni, che non richiedono modifiche.

  - × Problemi qualora fosse necessario rilevare dati da sensori sul dispositivo mobile (es: posizione tramite GPS).
    - soluzione: inviare alla cloudlet i dati percepiti; comporterebbe riduzione dell'autonomia e aumento dei tempi di risposta.
  - × Performances ridotte: unico processore nella cloudlet annulla il parallelismo derivante da esecuzione su diversi nodi Cloud.
  - × Complessità nella gestione del dispositivo clonato.

Una soluzione migliore

- Migliorie a livello di applicazione e non di piattaforma.
- Introduzione di **Elastic applications** [5], formate da più componenti chiamate **weblets**.
  - Ogni weblet può essere eseguita in locale sul dispositivo o sul Cloud.
  - La scelta è data dalle configurazioni dell'applicazione, stato del device, stato del Cloud, misurazioni di performance, etc .
  - Diverse modalità influenzano i criteri di scelta per una determinata applicazione: high-speed, power-saving, etc.
  - Possibilità di migrare dinamicamente l'esecuzione di weblets dal dispositivo al Cloud e viceversa.

Elastic Applications

- Partizionate in modo da avere weblets in esecuzione sul dispositivo e sul Cloud (modello Client/Server side delle web applications).
  Principi:
  - Organizzazione delegata all'utente: il Middleware deve fornire SDK necessario.
  - Posizione delle weblets trasparente all'applicazione.
  - Minimo accoppiamento fra diverse weblets.

- Configurazione di esecuzione (quindi "posizione" delle weblets) determinata al lancio e potenzialmente modificata in seguito, a runtime.
  - √ Grande flessibilità: è possibile seguire schemi per ottimizzare risparmio energetico, velocità di esecuzione, traffico di rete, etc.
  - √ Maggiore robustezza, tramite replicazione di weblets.

Elasticity Patterns

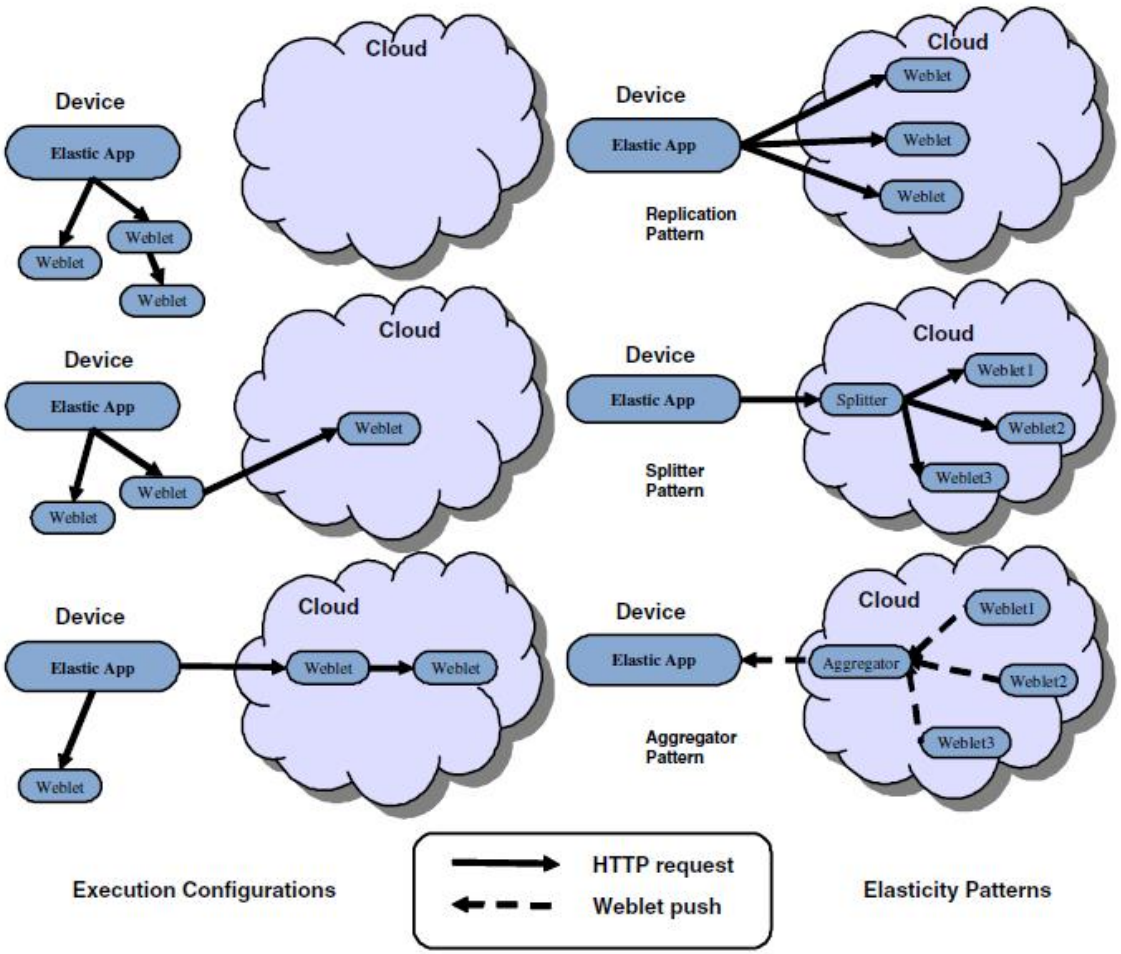
- Alcuni patterns per la collocazione di weblets sul dispositivo (client side) o Cloud (server side)

- **Replication pattern**: più weblets con la stessa interfaccia.
  Due diversi approcci:
  - Pools: richieste inviate ad una delle weblets in base alla sua disponibilità.
    - √ Permette politiche di load balancing.
    - √ Utile per eseguire operazioni su grandi set di dati.
  - Shadowing: richiesta inviata a più weblets in parallelo, la prima risposta ricevuta è utilizzata.
    - √ Maggiore tolleranza agli errori.

- **Splitter pattern**: unica weblet (splitter) smista richiesta a diverse worker weblets, le quali forniscono diverse implementazioni dell'unica interfaccia.
  - √ Migliora la user experience unificando diversi servizi in un'unica applicazione (es: diversi social networks).
  - √ Permette estensioni dell'applicazione che non ne modificano la struttura.

Elasticity Patterns (2)

- **Aggregator pattern**: aggregator weblet riceve dati da più worker weblets e li trasferisce al dispositivo con un meccanismo di push.

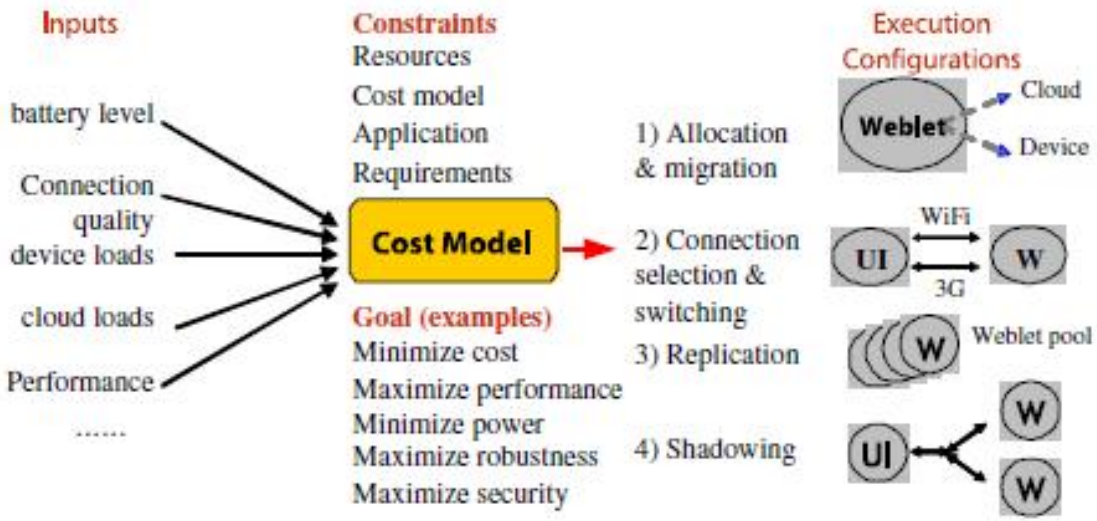


Weblets vs Web services

Weblets	Web services
HTTP (REST interface)	HTTP (REST or SOAP interface)
Single client	Many clients
Client is application root or other weblet	Clients are generally browsers or other web services
Short-lived and long-lived requests	Generally short-lived requests
Dynamic endpoints (may migrate)	Fixed endpoints
Lifetime is client dependent	Lifetime is client independent
Runs on servers or client (cloud or device)	Runs on servers
Push to client possible	Not available or non-standard

Modello di costo

- Ottimizzazioni per l'esecuzione della Elastic Application.
- Input: rilevazioni da dispositivo (carico complessivo, livello di batteria, condizioni della rete, ...) e Cloud (carico registrato).
- Output: azioni che favoriscono una configurazione di esecuzione ottimale (migrazione di weblets, allocazione di risorse sul Cloud, ...).



Obiettivi del modello

- Costo energetico: minimizzare il consumo di batteria.
  - Migrare più weblets sul Cloud, riducendo l'utilizzo del processore.
    - × Benefici ridotti dal maggiore utilizzo del modulo radio, utile solo per tasks complessi.
- Costo monetario: minimizzare il costo derivante dall'utilizzo della piattaforma Cloud.
  - Generalmente un provider misura il costo in base a cicli di CPU, memoria utilizzata e traffico di rete.
  - Valutazione per-weblet, considerando quelle su Cloud: tempo di esecuzione, memoria utilizzata, comunicazioni intra-Cloud, etc.
  - Migrare weblets più "costose" sul dispositivo.

Obiettivi del modello (2)

- Performances: massimizzare throughput e minimizzare tempi di risposta.
  - Occorre considerare latenza indotta da comunicazioni col Cloud, bandwidth del canale di comunicazione, cicli di CPU e memoria utilizzata dalla weblet.
  - Migrare weblets che fanno uso intensivo di CPU sul Cloud e quelle fortemente interattive sul dispositivo, considerando i vari fattori.
- Sicurezza e privacy: ridurre i rischi.
  - Proteggere dati sensibili dell'utente: contatti, informazioni su carte di credito, etc.
  - Mantenere sul dispositivo (impedire quindi la migrazione su Cloud) di weblets che operano su dati ritenuti confidenziali.

Utilizzo del modello di costo

- Algoritmo di machine learning sul dispositivo che determini la configurazione di esecuzione ottimale.
- Configurazione ottimale ricavata applicando una tecnica di Naive Bayesian Learning, considerando la precedenti:

$$y^* = argmax_y p(y) \prod_{i=1}^{L} p(x_i|y) \prod_{j=1}^{M} p(z_j|y)$$

  - x : vettore di valori che rappresentano lo stato del dispositivo (livello di batteria, banda in upload, ...).
  - z : vettore di preferenze utente.
  - y : esprime la configurazione precedente e ha valore compreso tra 1 e N (numero di possibili configurazioni).
    Ogni numero corrisponde ad una configurazione di esecuzione.

Architettura di riferimento

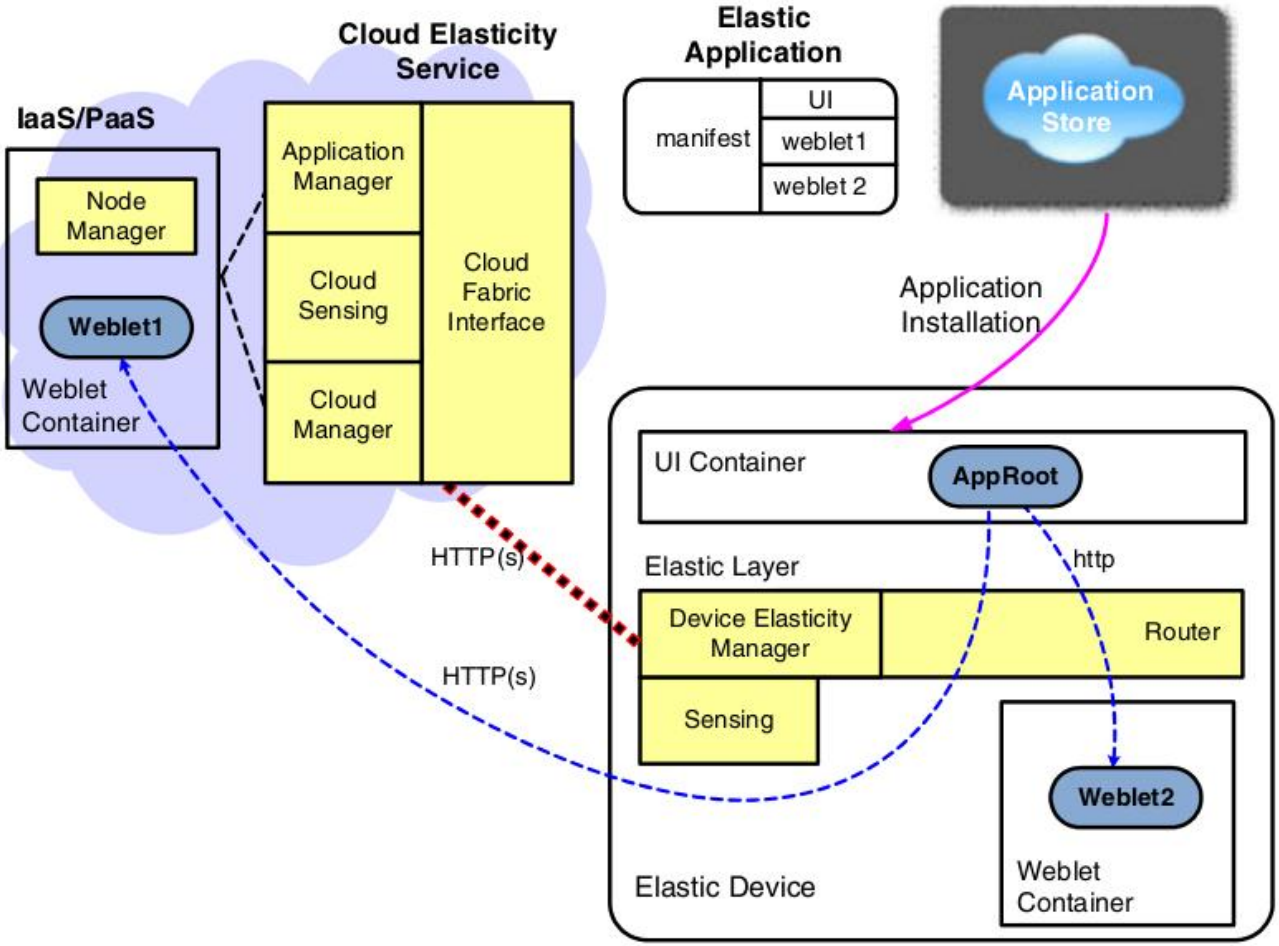
- Elastic application composta da:
  - Una User Interface.
  - Una o più weblets autonome, in esecuzione su dispositivo o Cloud, che mostrano interfacce **REST**ful.
  - Un manifest: XML di configurazione per l'applicazione e le singole weblets (contenente requisiti e restrizioni imposte dall'utente, firme digitali per autenticazione con la piattaforma Cloud, ...).

Elastic Applications: Componenti

- **Device Elasticity Manager** (DEM), collocato nel dispositivo.
  Responsabile della configurazione dell'applicazione e modifiche successive, nello specifico:
  - Posizione delle weblets.
  - Replicazione (con pool o shadowed) di weblets.
  - Selezione del percorso (inteso come interfaccia di rete: Wifi, 3G) per comunicare con le weblets su Cloud.
- La componente router si occupa di passare richieste HTTP dalla UI alle weblets, rendendo trasparente la loro posizione attuale (mascherando interruzioni qualora esse migrassero).
- Riceve dati relativi allo stato del dispositivo per determinare quando e dove istanziare nuove weblets.

Elastic Applications: Componenti (2)

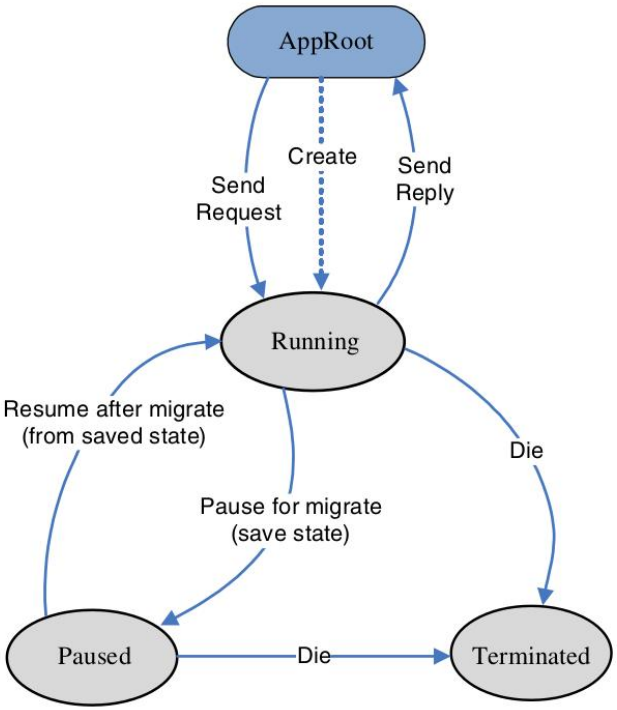
- **Cloud Elasticity Service** (CES), fornito dal provider per più dispositivi e collocato nella piattaforma Cloud.
  Si compone di:
  - Cloud manager: gestisce risorse dei nodi Cloud sottostanti.
  - Application manager: mantiene applicazioni dei dispositivi e lancia weblets su diversi nodi Cloud.
  - Cloud sensing: collezione di dati relativi allo stato della piattaforma Cloud (errori, disponibilità di risorse, ...).
  - Cloud fabric interface: web service messo a disposizione dei dispositivi, fornisce al DEM le informazioni per comunicare con le weblets e ne salva lo stato, qualora il DEM le interrompesse per migrarle.
- Ogni weblet su Cloud è associata ad un Node manager, il quale comunica con il Cloud e Application manager interni al CES.



Architettura per Elastic applications.

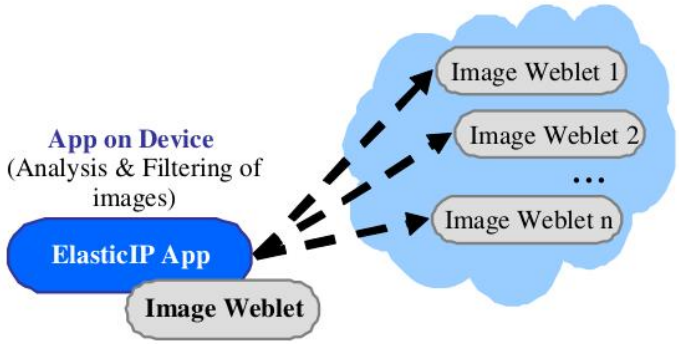
Weblets lifecycle

- L'**AppRoot** costituisce la UI ed invia le richieste HTTP alle weblets (passando per il DEM).
- Il SDK fornito permette di creare Elastic Applications usando linguaggi di alto livello.
  - Chiamate per creare una weblet, inviarle richieste e ricevere risposte da essa, sfruttando un meccanismo event-driven.



Applicazioni: Image processing

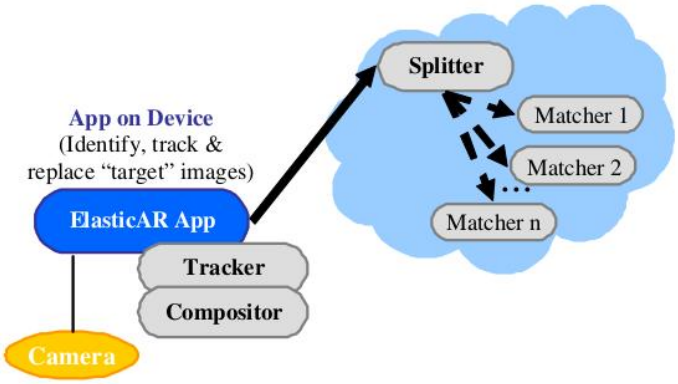
- Operazioni di filtro applicate ad un set di immagini.



- Replication pattern (pool) per processare immagini in parallelo.
- Possibilità di specificare:
  - Numero di immagini da processare in parallelo.
  - Numero di weblets da eseguire su Cloud.
  - Tipo di filtro da applicare alle immagini.

Applicazioni: Augmented reality

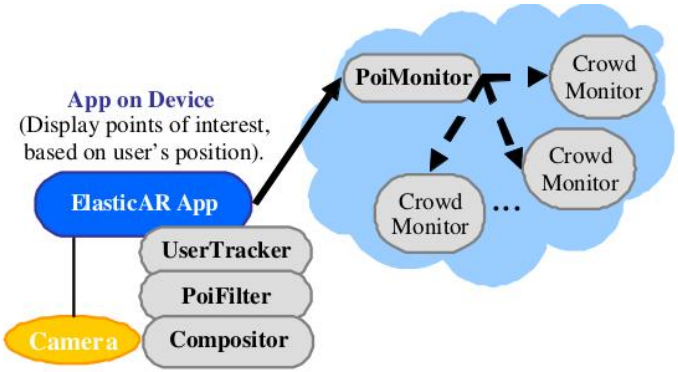
- Identificazione di oggetti attraverso fotocamera e alterazione del loro aspetto.



- Tracking (Tracker weblet) e rendering (Compositor weblet) effettuati sul dispositivo.
- Splitter pattern per assegnare diversi oggetti a diverse Matcher weblets, poste su Cloud.
- Le Matcher weblets si occupano di estrarre da un database le immagini associate ad ogni oggetto estratto, in base al linguaggio scelto dall'utente, sostituite nell'immagine dal Compositor weblet.

Applicazioni: Augmented reality (Video)

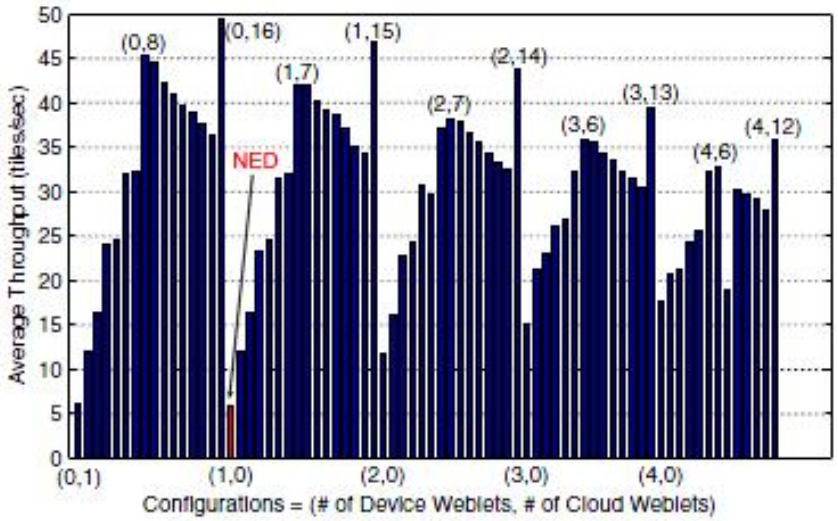
- Identificazione di punti di interesse (POIs/demos) e visualizzazione di informazioni su videocamera del dispositivo.

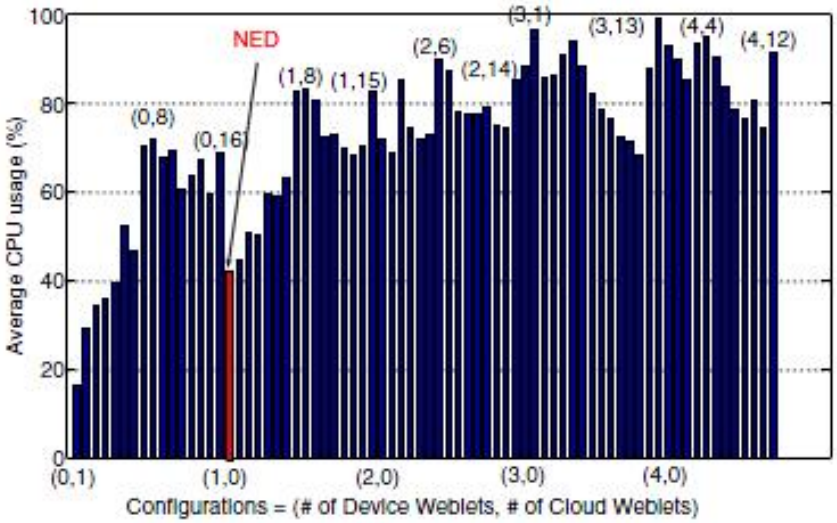


- Informazioni sui POIs nell'area ottenute ricavando posizione e direzione dell'utente (da weblet UserTracker).
- La weblet PoiFilter su dispositivo determina i POIs inquadrati dalla camera.
- Splitter pattern per ottenere informazioni sui POIs inquadrati (PoiMonitor weblet) ed il numero di persone nell'area.
  Le informazioni sono ottenute attraverso richiesta ad un web service su Cloud.
- Compositor weblet sovrappone le informazioni al video.

Elastic Applications: Prestazioni

- Applicazione di image processing su elastic device (ED) e non elastic device (NED), con diverse configurazioni di weblets.





Risultati di throughput ed utilizzo di CPU per 74 configurazioni diverse.

- Si può notare come per configurazioni che usano 2 weblet sul dispositivo, avviarne in numero maggiore sul Cloud aumenta il carico di CPU.

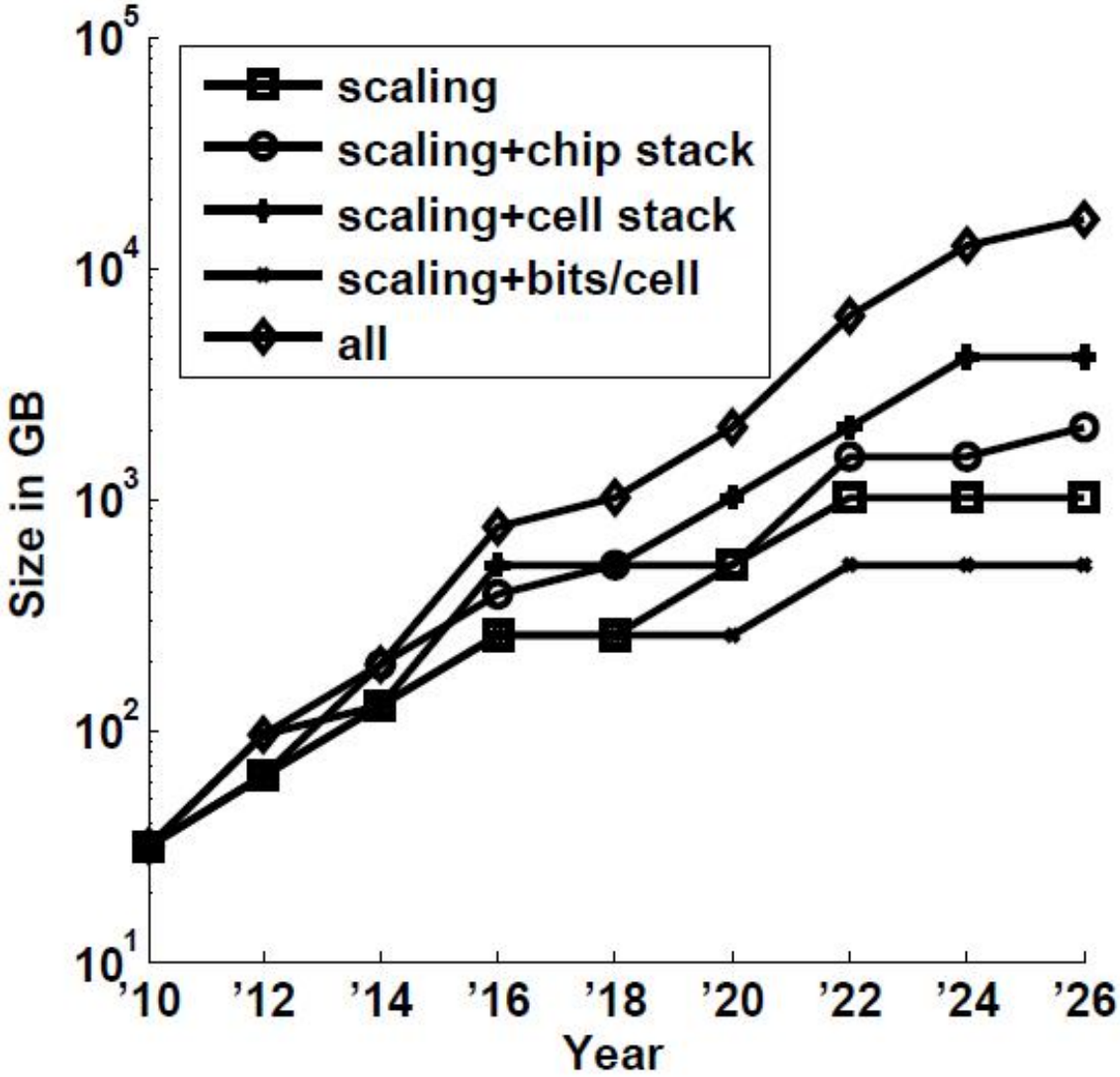
Connettività dei dispositivi mobili

- L'accesso a servizi Cloud da dispositivi mobili soffre dei limiti imposti dalle tecnologie utilizzate:
  - × Alta latenza e scarsa robustezza (dai 3 ai 10 secondi per ricevere i risultati di una query da un motore di ricerca, con connessione 3G).
  - × Alti tempi di risposta, nonostante l'aumento del throughput con tecnologie più recenti (reti 4G), causati dalla tendenza degli utenti a scambiare piccoli pacchetti di dati ed al tempo necessario all'antenna per svegliarsi dalla modalità standby (1-2 secondi).
  - × Alto consumo di energia, derivante dall'utilizzo di reti cellulari a lungo raggio, riduce l'autonomia della batteria.

Capacità di memoria dei dispositivi mobili

- Il gap nella capacità di memoria primaria/secondaria tra dispositivi mobili e desktop si sta progressivamente assottigliando (fino a 64 GB di memoria flash nei dispositivi recenti).
- Si prevedono migliorie alla tecnologia Flash (o altre tipologie di **N**on **V**olatile **M**emory) nei prossimi anni, in termini di capacità (data da numero di celle, bits per cella, ...).

	Flash				Other NVM technology				
year	'10	'12	'14	'16	'18	'20	'22	'24	'26
tech (nm)	32	22	16	11	11	8	5	5	5
scaling factor	1	2	4	8	8	16	32	32	32
chip stack	4	4	6	6	8	8	12	12	16
cell layers	1	1	1	2	2	4	4	8	8
bits per cell	2	3	2	2	2	1	1	1	1



Previsioni sull'evoluzione nella capacità di memoria per smartphones di fascia alta.

Pocket Cloudlets

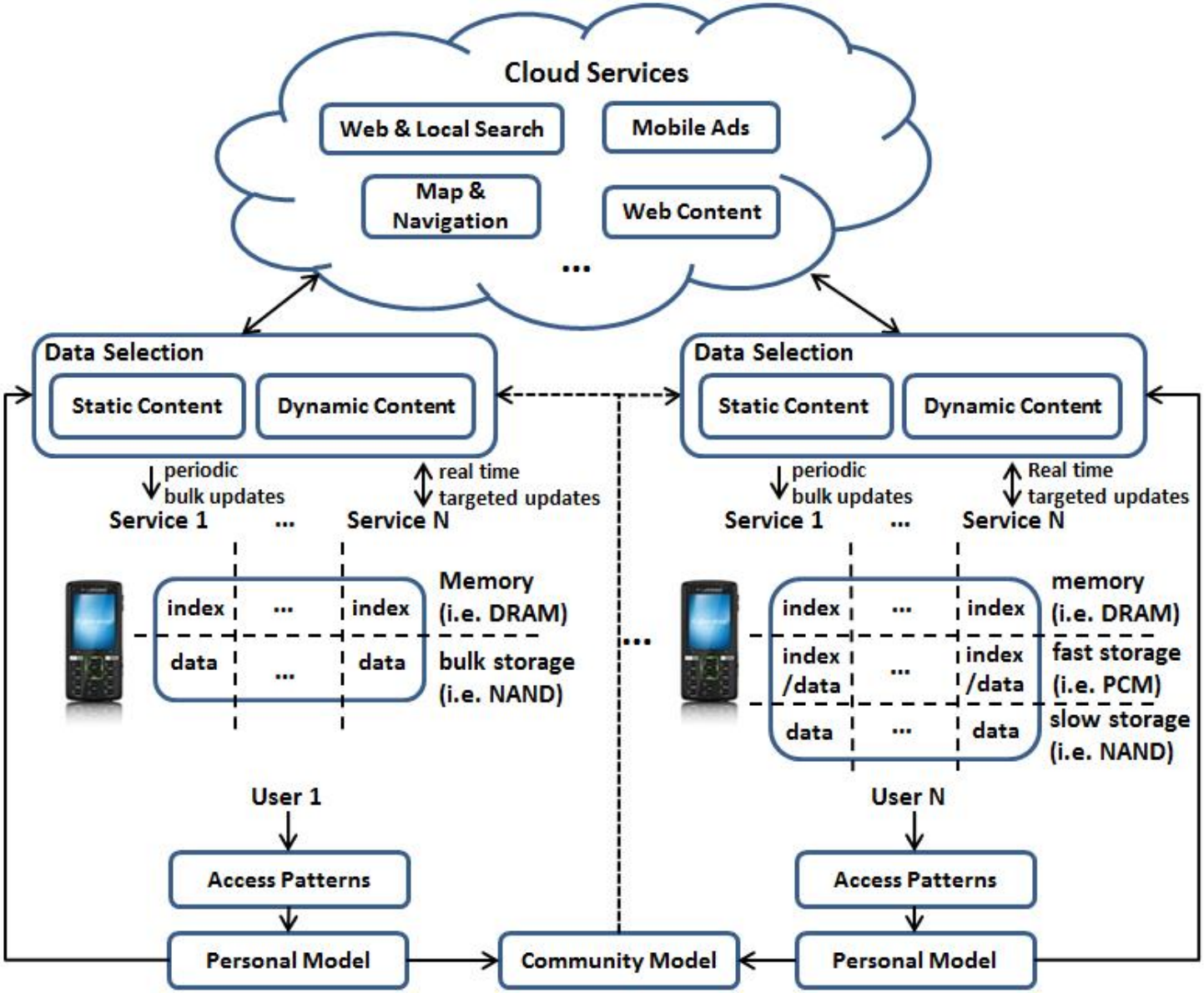
- Sfruttare l'aumento nelle capacità di memorizzazione per spostare parti specifiche di un servizio Cloud (o sua interezza) sulla memoria del dispositivo: **Pocket Cloudlets** [3].
- Benefici:
  - √ L'accesso istantaneo alle informazioni (già presenti sul dispositivo) elimina latenza e consumo energetico derivanti dalle comunicazioni col Cloud.
  - √ Riduzione del carico complessivo delle reti cellulari (che si prevedono essere una risorsa critica, con la maggiore diffusione dei dispositivi mobili).
  - √ Personalizzazione facilitata dei servizi in base ai pattern di utilizzo di singoli utenti.
  - √ Possibilità di conservare informazioni personali nelle cloudlets e proteggere la privacy degli utenti.

Gestione di Pocket Cloudlet

- Criteri di **selezione** dei dati da conservare in NVM per un dato servizio Cloud.
  - Modello personale costruito a partire dai pattern di accesso di un singolo utente al servizio.
  - Modello comunitario costruito dal Cloud provider in base a più modelli personali, per identificare le componenti di maggiore utilizzo del servizio.
- Frequenza di aggiornamento dei dati.
  - Può basarsi sulla bandwidth disponibile (Wifi vs 3G) o sullo stato di carica del dispositivo.
    - ✓ Efficiente per grandi quantità di dati statici: tiles di una mappa visualizzata.
    - × Inefficiente su dati non statici: news, e-commerce, etc. Per questi ultimi occorrono aggiornamenti in real time.

Gestione di Pocket Cloudlet (2)

- Utilizzo di NVM (es: tecnologia NAND flash) per memorizzazione stabile di grandi quantità di dati relativi al servizio Cloud.
- Utilizzo di memoria veloce volatile (es: DRAM, memoria primaria di uno smartphone) per indicizzare i dati su NVM, permettendo un accesso più efficiente.
- Possibile aggiunta di un terzo layer intermedio di NVM PCM (Phase-change memory, più veloce di NAND flash) per memorizzare gli indici, disponibili così da boot time. Lo strato DRAM può essere utilizzato per dati acceduti più frequentemente.



Supporto architetturale alle Pocket Cloudlets.

PocketSearch

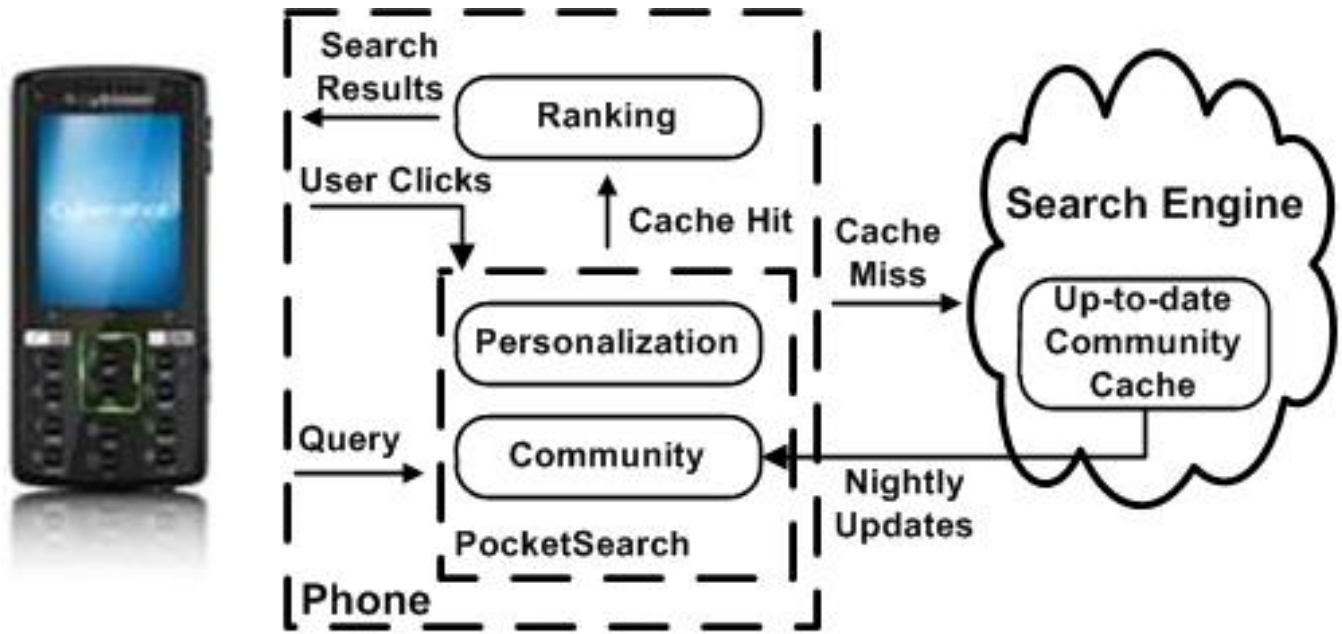
- Applicazione per mobile search che immagazzina risultati delle queries nella memoria del dispositivo, consentendo di non usare la connessione 3G.

- Fornisce un servizio "offline" di auto-completamento nella ricerca di contenuti, memorizzando i risultati di una ricerca precedente e non il contenuto web degli stessi.

PocketSearch: Componenti

- La cache conservata distingue due componenti:
- **Community cache**: conserva un piccolo insieme di queries/risultati più popolari tra gli utenti.
  - Informazioni estratte dai logs di ricerca e aggiornate mentre il dispositivo è in carica.
  - Forniscono una cache di partenza basata sui trend attuali.
- **Personalization cache**: gestisce le queries dell'utilizzatore.
  - Memorizza gli ultimi risultati ottenuti.
  - Conserva una classifica dei risultati di ricerca più selezionati dall'utente.

PocketSearch: Architettura



Architettura dell'applicazione PocketSearch.

PocketSearch: Generazione della cache

- I risultati di ricerche contenuti nella cache sono estratti dai logs come triple $\langle query,\ search\ result,\ volume \rangle$, ordinate per l'attributo *volume*.

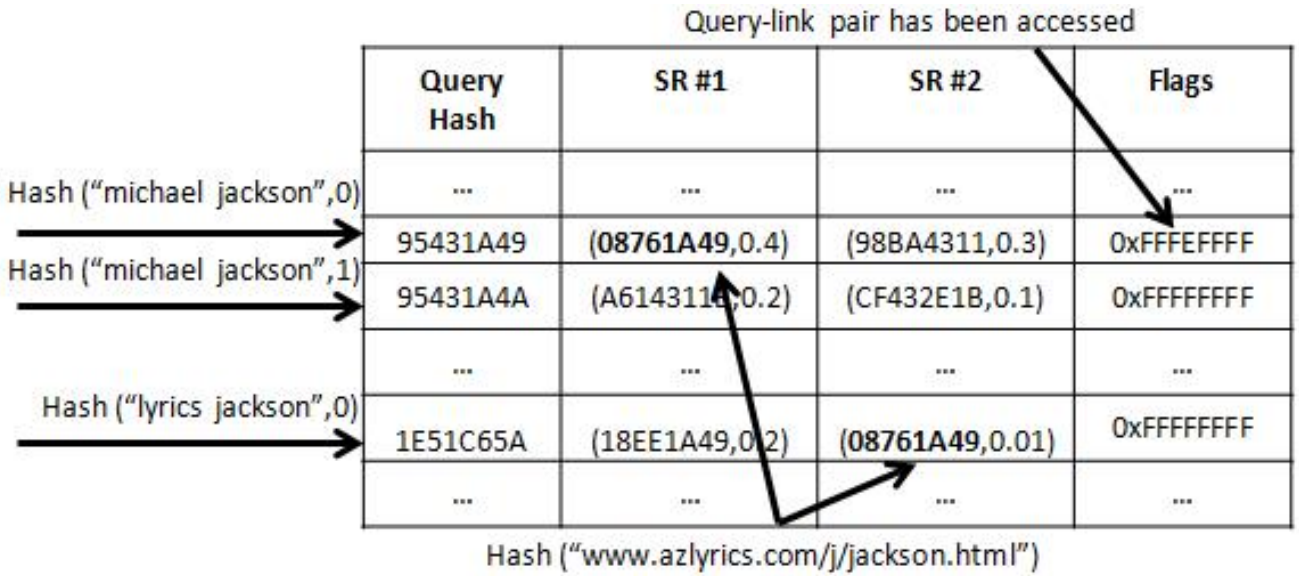
- Il numero di record da aggiungere alla cache è selezionato in base a due criteri:
  - Memory Threshold (flash o DRAM): si aggiungono coppie $\langle query,\ search\ result \rangle$ fino a che una soglia di capacità M_{th} non è raggiunta.
  - Cache Saturation Threshold: si aggiungono coppie fino a raggiungerne una con volume normalizzato (divisione del volume della coppia per quello di tutte le coppie nei search logs) minore di una soglia V_{th}.

PocketSearch: Strutture di memorizzazione

- Sono necessari efficienti meccanismi di memorizzazione della cache.
  - Minima quantità di memoria utilizzata per conservare i risultati di ricerche.
  - Minimo tempo di accesso ai risultati, ranking e visualizzazione degli stessi.
- Due componenti:
  - Hash Table in memoria principale: collega le queries ai loro risultati (dimensione approssimativa: 200 KB).
  - Database su memoria flash: memorizza i risultati delle queries (dimensione approssimativa: 1 MB).

PocketSearch: Hash Table

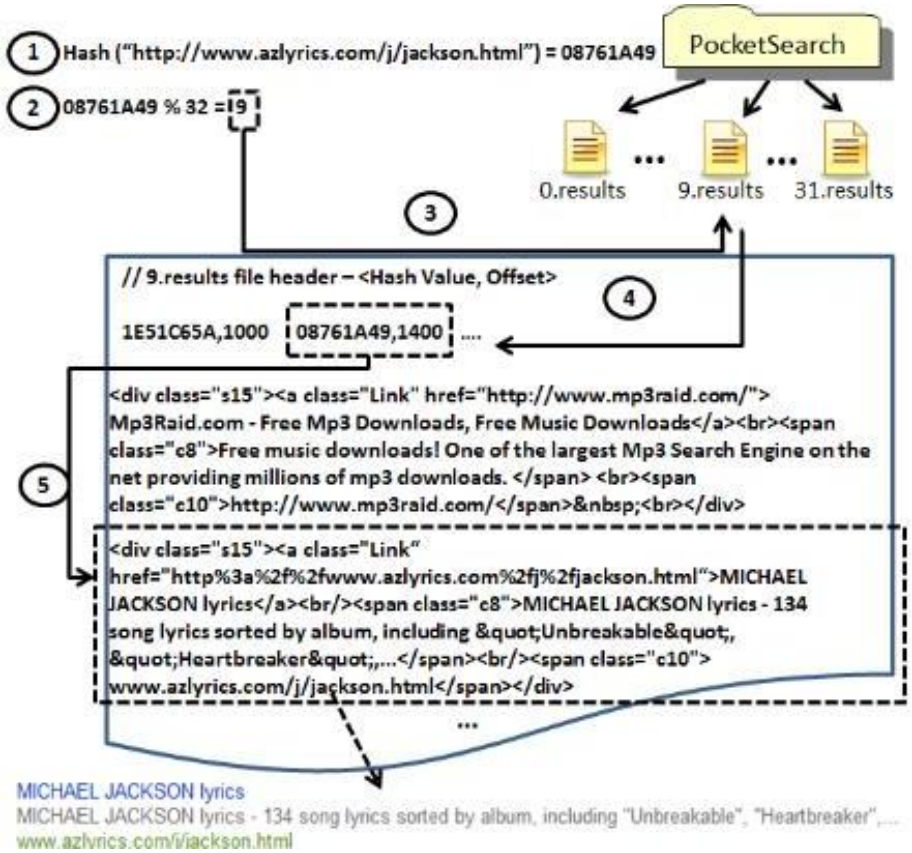
- Data una query, consente ricerca in tempo $O(1)$.
  - Cache hit: puntatore ai risultato della ricerca nel database e rispettivi ranks.
  - Cache miss: avvio della ricerca su motore web.

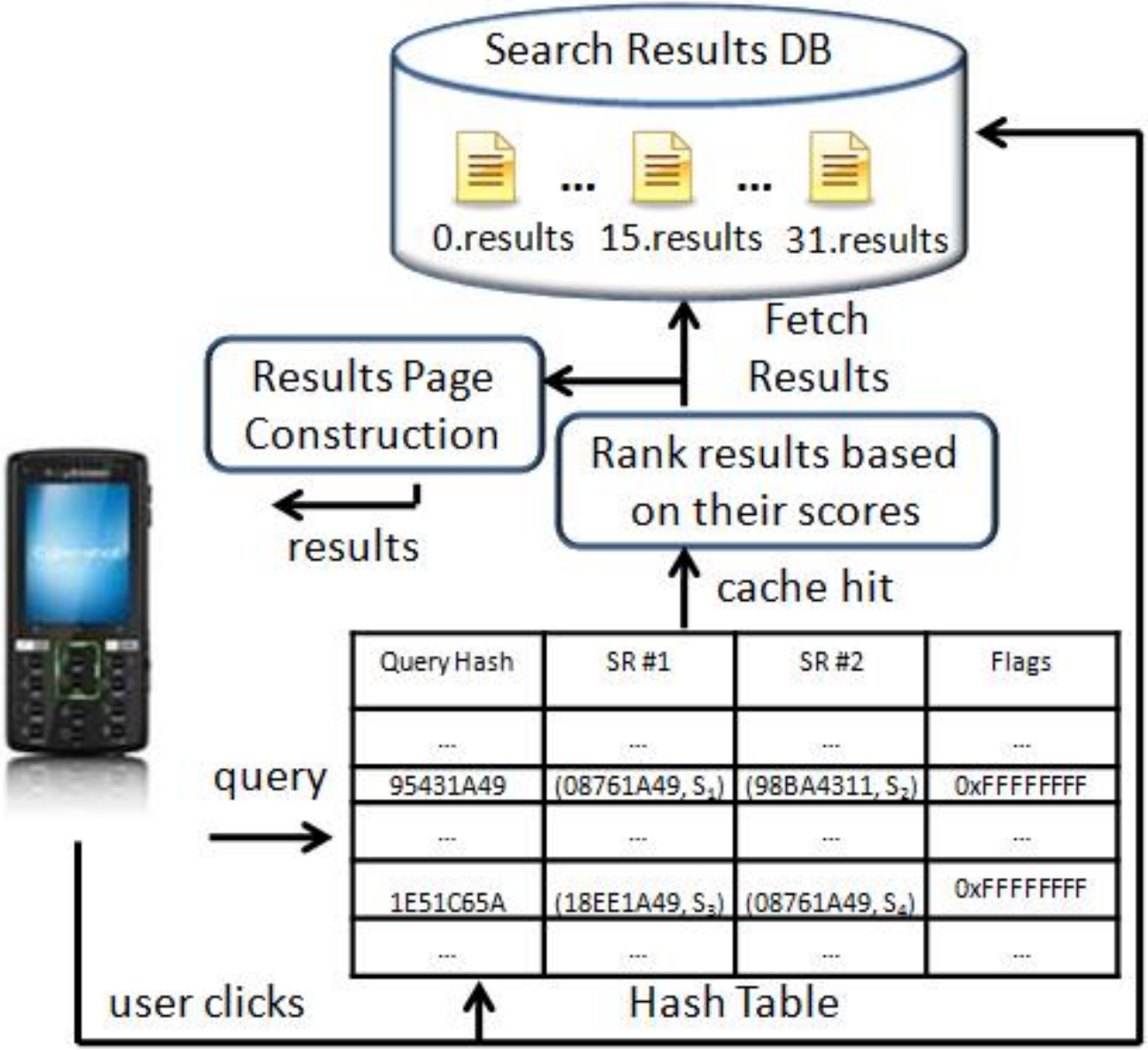


- Per ogni query string contiene:
  - Hash value della query string.
  - Puntatori ai primi due risultati della ricerca, comprensivi di rank.
  - Flags a 64 bit per informazioni aggiuntive.

PocketSearch: Database

- Risultati di ricerca memorizzati su flash in un database di file testuali, per facilitarne la portabilità.
  - Contiene URL, descrizione e versione human-readable dell'indirizzo di ogni risultato ottenuto.
  - Per evitare frammentazione, ogni file contiene più risultati (occupanti ognuno 500 bytes).

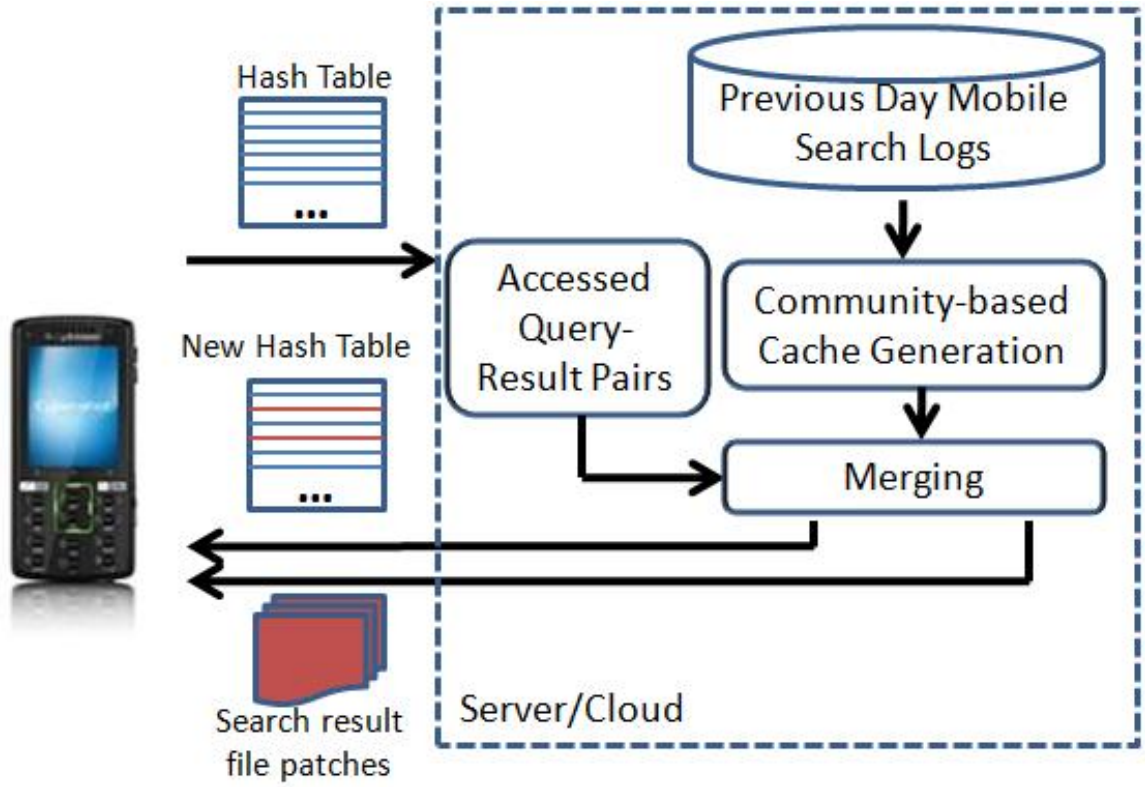




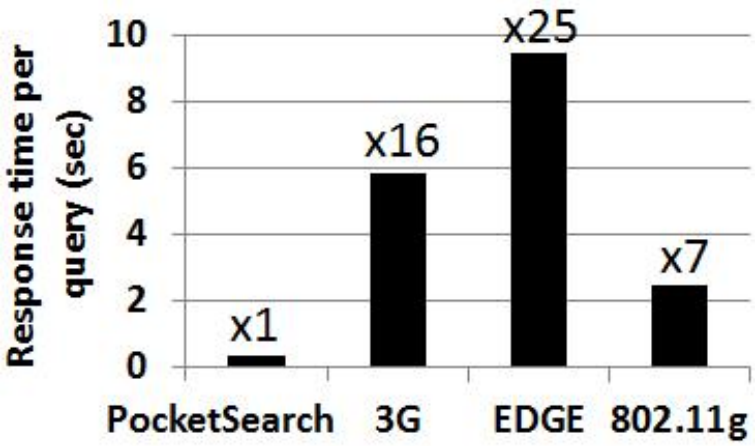
Strutture dati dell'applicazione PocketSearch.

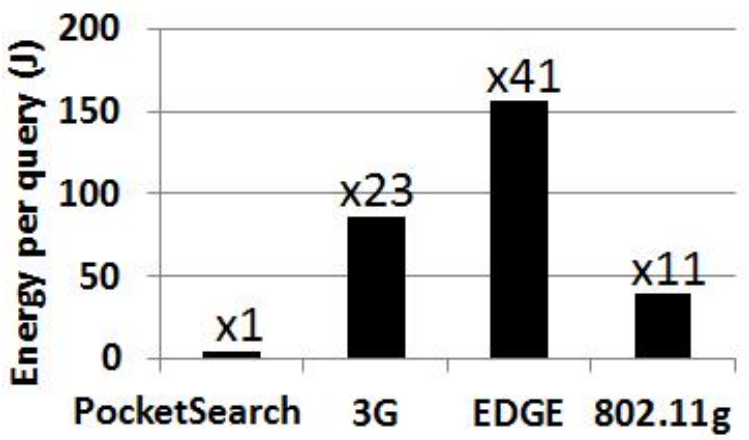
PocketSearch: Il Cloud

- Il Server su Cloud effettua le seguenti operazioni:
  - Riceve dal dispositivo la versione attuale della Hash Table e rimuove le entry ignorate dall'utente (usando il campo Flags) e quelle con il rank inferiore ad una certa soglia.
  - Estrae periodicamente le queries e i risultati più popolari dai logs e li aggiunge alla table (preferendo quelle con rank più alto, in caso di conflitti).
  - Invia la nuova hash table e le necessarie patches per il database all'utente.



PocketSearch: Prestazioni





Comparazione tra cache hit e miss per tempo e consumo energetico.

Operation	Average Time (ms)	Percentage
Hash Table Lookup	0.01	$\approx 0\%$
Fetch Search Results	10	2.7%
Browser Rendering	361	96.7%
Miscellaneous	7	1.7%
Total	378	100%

PocketSearch: Conclusioni

- √ Risparmio di risorse per il dispositivo, in termini di banda di rete utilizzata e autonomia di batteria.
- √ Riduzione del carico di reti cellulari e datacenters, gestendo in locale parte delle queries.
- √ Possibilità di esporre il servizio ad altre cloudlets su dispositivo.

- × Contesa di memoria primaria, per memorizzazione di indici, con altre applicazioni sul sistema: potenziale peggioramento delle performances in seguito a spostamento su flash.
- × Necessità di politiche di sicurezza per separare le caches di diverse applicazioni.

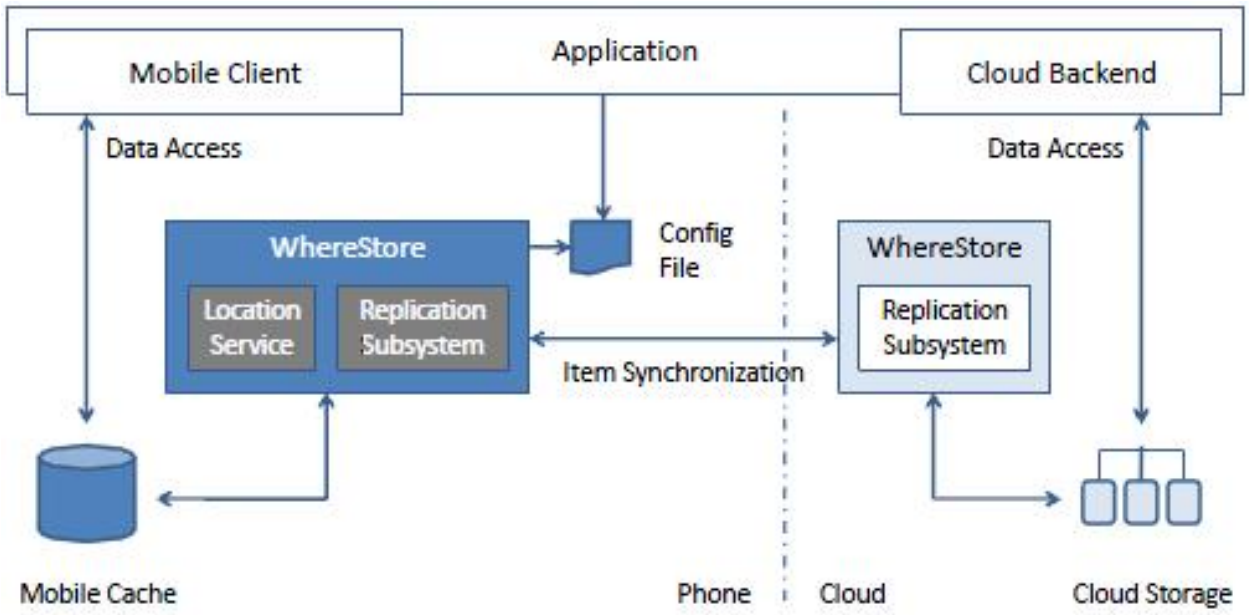
WhereStore

- Assunzioni:
  - Molte applicazioni sono utilizzate in modo specifico in base alla posizione dell'utente (agenda lavorativa in ufficio, applicazione per le news sul traffico in auto, ...).
  - La precedente posizione di un utente è una buona indicazione sulle future e quindi sulle potenziali necessità di informazioni.

- Utilizzare uno storico di posizioni del dispositivo per determinare quali dati replicare localmente dal Cloud, in un datacenter accessibile a più applicazioni: **WhereStore** [4].

WhereStore: Componenti

- Sistema client/server, localizzati rispettivamente su dispositivo mobile e Cloud.
- Il client si compone di due elementi fondamentali:
  - Un location service: fornisce informazioni circa le possibili future posizioni del dispositivo mobile.
  - Un replication subsystem: mantiene una cache locale di dati, sincronizzata con il Cloud.

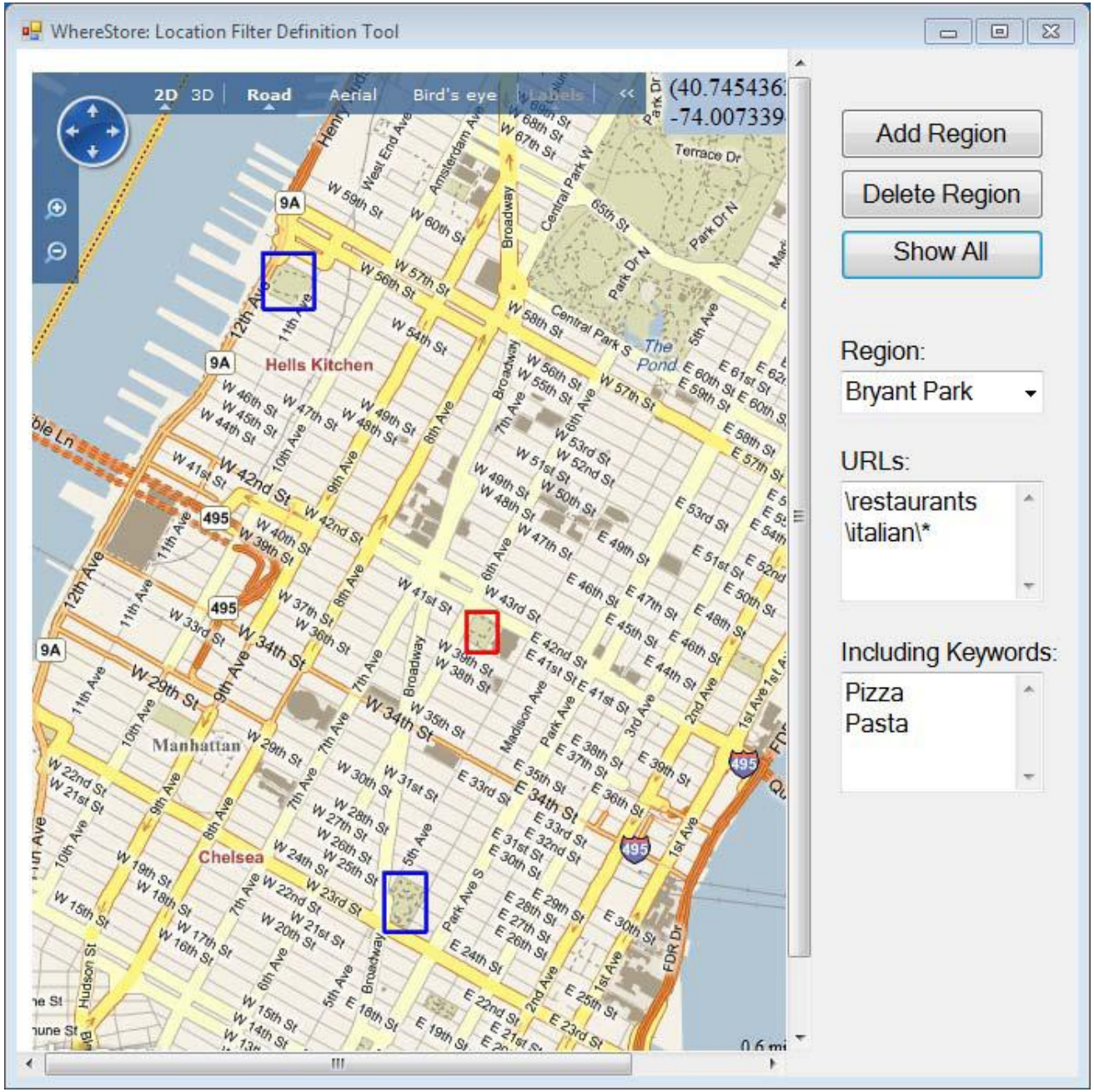


WhereStore: Replication subsystem

- Il replication subsystem si occupa di mantenere repliche dei dati per ogni applicazione, su dispositivo e sul Cloud.
- Prevede i concetti di *gruppo* e *regione* tra i metadati associati ad un oggetto, per identificare rispettivamente il tipo di dato (files di testo, musica, ...) ed il suo contesto di utilizzo (casa, ufficio, ...).
- Ogni replica su dispositivo consiste in un filtro: un predicato sui metadati di uno o più oggetti nella collezione (es: immagini JPEG con geo-tag relativo ad una certa regione).
- Ad ogni replica è anche associato un limite massimo di storage da occupare.

WhereStore: Location service

- Il location service, sfruttando uno storico degli spostamenti registrati con relativi timestamp, fornisce l'insieme $(l_1, l_2, \ldots, l_n)$ di possibili luoghi in cui l'utente si verrà a trovare, associando ad ogni luogo l_i una probabilità p_i.
- Per generare il set di posizioni e mantenere lo storico di quelle passate, viene utilizzato il servizio Cloud StarTrack.
- Per ogni l_i viene creato un filtro f_i, considerando tutte le regioni che includono l_i. Il set viene ricalcolato ogni volta che l'utente oltrepassa i confini di aree predefinite.



Tool per definire i filtri utilizzati.

WhereStore: Il Cloud

- Sincronizzazioni periodiche avvengono fra il dispositivo ed il Cloud.
- La sincronizzazione prevede le seguenti operazioni:
  - Il Cloud riceve il set di filtri relativo alla posizione attuale (e possibili future) del dispositivo, con relative probabilità.
  - Vengono selezionati gli oggetti relativi ad ogni filtro f_i dallo storage del Cloud.
  - Gli oggetti sono ordinati in base al loro rank, calcolato come $p_i \times k_j$ (probabilità del relativo filtro per un valore di priorità associato all'oggetto).
  - Solo i primi n oggetti, le cui dimensioni totali non superano una soglia predefinita, sono inviati al dispositivo.

Conclusioni

- Argomenti trattati:
  - Vantaggi e potenziali utilizzi dati dalla combinazione di dispositivi mobili smart, Cloud computing ed interfacce RESTful.
  - Context-aware applications e Cloud.
  - Ottimizzazione delle interazioni tra dispositivi mobili e Cloud mediante meccanismi di caching su NVM, eventualmente con criteri location-based.
  - Distribuzione delle componenti di un applicazione tra dispositivo e Cloud: Elastic applications.
- Prospettiva futura:
  - Un'interazione più efficiente tra dispositivi mobili e piattaforme Cloud, quindi tra utenti e servizi, grazie a migliorie nelle capacità dei singoli dispositivi e nelle infrastrutture di reti cellulari utilizzate.

Riferimenti I


Lucia Terrenghi, Thomas Lang, and Bernhard Lehner. 2009.
Elastic mobility: stretching interaction.
In Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '09). ACM, New York, NY, USA.


Jason H. Christensen. 2009.
Using RESTful web-services and cloud computing to create next generation mobile applications.
In Proceeding of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications (OOPSLA '09). ACM, New York, NY, USA.


Emmanouil Koukoumidis, Dimitrios Lymberopoulos, Karin Strauss, Jie Liu, and Doug Burger. 2011.
Pocket cloudlets.
In Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems (ASPLOS '11). ACM, New York, NY, USA.

Riferimenti II


Patrick Stuedi, Iqbal Mohomed, and Doug Terry. 2010.
WhereStore: location-based data storage for mobile devices interacting with the cloud.
Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond (MCS '10). ACM, New York, NY, USA.


Xinwen Zhang, Anugeetha Kunjithapatham, Sangoh Jeong, and Simon Gibbs. 2011.
Towards an Elastic Application Model for Augmenting the Computing Capabilities of Mobile Devices with Cloud Computing.
Mob. Netw. Appl. 16.


Louis Columbus. July 24, 2011.
Predicting Cloud Computing Adoption Rates.
http://softwarestrategiesblog.com/2011/07/24/predicting-cloud-computing-adoption-rates/.

Riferimenti III


Stewart Harper. September 15, 2011.
GIS Developers: Mobile is Here. Time to Build for It.
http://blog.safe.com/2011/09/gis-developers-mobile-is-here-time-to-build-for-it/.