Energy-Aware Keyword Search on Mobile Phones

Weixiong Rao
Department of Computer Science
University of Helsinki, Finland
weixiong.rao@cs.helsinki.fi

Kai Zhao
Department of Computer Science
University of Helsinki, Finland
kai.zhao@cs.helsinki.fi

Eemil Lagerspetz
Department of Computer Science
University of Helsinki, Finland
eemil.lagerspetz@cs.helsinki.fi

Pan Hui
Deutsche Telekom Laboratories
German
ben@net.t-labs.tu-berlin.de

Sasu Tarkoma
Department of Computer Science
University of Helsinki, Finland
sasu.tarkoma@helsinki.fi

ABSTRACT

With the explosive growth of communication technologies, modern mobile phones become more powerful than ever. Unfortunately, the battery lifetime of mobile phones is still limited, and energy awareness is a priority of designing applications on mobile phones. To demonstrate the energy awareness on mobile phones, we propose a new approach for full-text keyword searches to retrieve content matching input keywords. The proposed approach generalizes the two solutions that answer the keyword searches either (i) locally by mobile phones themselves or (ii) remotely by power servers when the keywords are offloaded to such servers. Instead, for low energy, we propose to split the keywords in a search into two subsets, such that one subset is processed by the mobile phone and another by the remote server. Our preliminary experiment results indicate the hybrid approach can save the most energy for the keyword searches than the local and remote approaches. In our example tests, energy of the hybrid approach saves 75% and 47% of energy when compared to local execution and complete offloading approaches, respectively.

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Performance evaluation

Keywords

Keyword Search, Mobile Phones, Energy

1. INTRODUCTION

With the explosive growth of communication technologies, modern mobile phones become more powerful than ever. The recent Apple iPhone 4S is equipped with an 800 MHz dual-core Apple A5 processor, 64 GB storage, and 512 MB RAM [1]. With the powerful mobile phones, users access various Internet services, such as weather, instant messaging, social media, Internet TV, document processing.

Unfortunately, the battery lifetime of mobile phones is still limited, and energy awareness has become a priority of designing applications on mobile phones. To demonstrate the energy awareness on mobile phones, in this paper, we propose a new approach for the full-text keyword search to retrieve content matching input keywords. Though iPhone and Android mobile search applications offer keyword search functions, such applications search only content titles or simple text files. Instead, we target the full-text search over various content types, including pdf files, web content, etc.

To answer the keyword searches, an optional solution, namely *local solution*, is that mobile phones locally process the searches and answer the query results. However, the retrieval of content on mobile phones may consume high energy. To save the energy for the keyword searches, an alternative solution, namely *remote solution*, is to offload the query q (consisting of the input keywords) from mobile phones to remote powerful servers or even cloud services via network communication (e.g., 3G networks, WiFi). The remote server then processes the query q and returns the query results back to the mobile phones. This offloading approach involves the tradeoff between (i) the decrease of the energy for processing q that is now executed by the remote server and (ii) the increase of the energy for offloading q and receiving the query results plus the the energy for being idle when waiting for the query results.

Nevertheless, we believe that the above two approaches may not be optimal solutions for the keyword searches. To this end, we propose a new hybrid approach by splitting the keywords in a query q into two subsets, q_s and $q \setminus q_s$, respectively. Then, we explore the possibility of offloading only the subset q_s to the remote server and meanwhile answering the subset $q \setminus q_s$ on the mobile phones. The union of the query results from both the mobile phones and remote servers is the final result of q. In this way, we expect that energy used by the hybrid approach is smaller than the above two approaches. In reality, the hybrid solution is a generalized framework. For example, if $q_s = \emptyset$ and $q \setminus q_s = f$, the whole query q is then locally answered by the mobile phone, and this is equivalent to the local approach; if $q_s = q$ and $q \setminus q_s = \emptyset$, the whole query q is then completely answered by the remote server. In view of this, the key is to find the subsets q_s and $q \setminus q_s$ in order to minimize the overall energy consumption for answering the whole query q.

The rest of this paper is organized as follows. Section 2 first

MCC'12, August 17, 2012, Helsinki, Finland.

gives an overview of the proposed solution. Next, Section 3 introduces the energy consumption equations of the proposed solution, and Section 4 focuses on designing algorithms to find the optimal subsets for minimal energy. Section 5 evaluates preliminary results for the proposed scheme. After that, Section 6 investigates related work, and Section 7 finally concludes the paper and discusses the forthcoming works.

2. OVERVIEW

2.1 Problem Statement

For a mobile phone running the keyword search application, we optimize the energy saving for the query engine of the search application under the following assumptions. First, we focus on the scenario where the users are moving around and battery recharge is unavailable. Otherwise, when users stay at home or office, they can conveniently recharge batteries, without energy concerns for file searches and other operations. Second, we assume that the mobile phone is used to store personal files and the files are not updated very frequently. Thus, via daily synchronization, a remote server maintains the exactly same filters (and the associated index files) as the mobile phone does. Finally, the keyword-based search can be answered (i) by the local mobile phone, or (ii) by the remote server. The remote server answers the keyword searches offloaded from the mobile phone via the widely available network channels, such as WiFi hotspots and 3G networks. However, the energy consumption of these two approaches differs and depends on many factors, such as the query conditions, query results, the network channels between the mobile phone and remote server.

Based on the above assumptions, our problem is to minimize the overall battery consumptions to answer N queries q_i with $1 \leq i \leq N$. Alternatively, its dual problem is to maximize the number of the answered queries when the battery capacity is predefined. For each query q_i, it consists of $|q_i|$ input terms (i.e., keywords), and supports a *boolean-based query model*: among all M content files, a query q_i retrieves all files containing one or multiple input terms in q_i. Intuitively, the query results of q_i is treated as the union over the query results of all input terms in q_i. The boolean-based query model can be extended to more advanced models, such as the threshold-based or top-k models [10].

2.2 Solution Overview

To answer the query q, the local approach, via the keyword search application on a mobile phone, directly retrieves document identifiers from the local index file (e.g., an inverted list to index the content) for q. Instead, for the remote approach, the query q is offloaded to the remote server and then waits for the processed results. After receiving the query q, the remote server processes q and returns the query results to the mobile phone.

Though the above two approaches work, they are not energy efficient. Before giving the hybrid approach, we first give a lemma for the boolean-based query model as follows.

Lemma 1 *Given two queries q and q_s, suppose that the terms in q are the superset of the terms in q_s. Then for a set of given documents, the query results for q are the superset of the results for q_s.*

We can easily derive the above lemma by the definition of the boolean-based query, which uses the OR operation to get the query results for all terms in the a query. The lemma ensures the proposed hybrid approach can correctly return the query result of q (i.e., no content is falsely missed). That is, the query q is spit into two parts:

Symbol	Meaning
q, q_s,$q \setminus q_s$	the query q, the subset of keywords in q offloaded to a remote server, and the remaining subset answered by a mobile phone.
$E_l(q)$,$E_r(q)$,$E_h(q)$	Energy consumption of a mobile phone to answer q by the local, remote and hybrid approaches.
$R(q)$,$R(t_i)$,$R(q_s)$	# of computation operations to answer q, the query consisting of term t_i, and the query consisting of q_s.
s_l,P_l (resp. s_r,P_r)	Seconds and watts used by a mobile phone (resp. a remote server) for computation operation
$D_{up}(q)$,B_{up} (resp. $D_{dn}(q)$,B_{dn})	Bytes to upload q (resp. download the results of q) and the bandwidth of the uploading (resp. downloading).

Table 1: Main used symbols and meanings

(i) a subset of the keywords in q, denoted by q_s, is offloaded and then answered by the remote server, and (ii) the remainder of the query $q \setminus q_s$ is directly executed by local mobile phones. The union of the query results from both the mobile phone and remote server is the final result of q.

For the proposed hybrid approach, the key is to adaptively split the query q into two parts q_s and $q \setminus q_s$ for the optimal energy consumption. We formulate the adaptive split of q into two subsets of queries, denoted by SSQ, as a min-cost flow graph in order to minimize the energy consumption. The split of the queries depends upon the solver to answer the SSQ problem. Due to the energy limit of mobile phones, during the running time of the keyword searches ,we avoid solving the SSQ problem for each input query q by mobile phones themselves. Instead, with query logs of end users as the input of the SSQ problem, the remote powerful server periodically solves the SSQ problem in an offline manner. After downloading the results of SSQ, the mobile phones then split each q into two subsets q_s and $q \setminus q_s$. The behind rationale is that end users are typically "lazy" and would not very frequently change their personal habits [7]

The following sections are organized as follows. First, Section 3 presents more details of the energy for the three approaches to answer queries. Then, Section 4 solves the SSQ problem. Table 1 summarizes the mainly used symbols and the associated meanings in Sections 3 and 4.

3. QUERY PROCESS OFFLOADING

In this section, we give the details of the energy consumption for three proposed approaches to answer queries.

3.1 Energy for the Local Approach

First, we assume that answering a query q needs to retrieve the number $R(q)$ of document candidates for the final query results. In terms of the full-text keyword searches, we define the *computation operation* to be the one for retrieving one candidate and then deciding that the candidate is a final query result or not. For the studied boolean query model in this paper, a document candidate, if containing any input keyword of q, is the final result, and the query q needs $R(q)$ computation operations.

We denote by $E_l(q)$ the energy of the mobile phone for the local approach to answer q. Suppose that on average the mobile phone spends s_l seconds for each query computation operation and consumes p_l watts per query computation operation. Then, $E_l(q)$ is computed by the multiplication of the used time, $s_l \cdot R(q)$, and consumed watts, $p_l \cdot R(q)$:

$$E_l(q) = [s_l \cdot R(q)][p_l \cdot R(q)] = s_l \cdot p_l \cdot R^2(q) \quad (1)$$

3.2 Energy for the Remote Approach

Next, we denote by $E_r(q)$ the energy consumption of the mobile phone for the remote approach to answer the query q. $E_r(q)$ contains two parts as follows.

- The energy of the mobile phone for being idle to wait for the processing results from the remote server. Suppose the remote server on average needs s_r seconds for each query computation operation, and totally $s_r \cdot R(q)$ seconds to process q. During the period, the mobile phone is idle, and the energy consumption is $s_r \cdot P_r \cdot R^2(q)$ watts (where p_r is the average amount of watts of the mobile phone for being idle per second).
- Besides, to enable the process of q on the remote server, the client and remote server need to exchange data. Denote $D_{up}(q)$ and $D_{dn}(q)$ to be the bytes of uploading the queries and downloading the query results, respectively, and B_{up} and B_{dn} the uploading and downloading network bandwidth, respectively. Then the time used for such uploading and downloading is $D_{up}(q)/B_{up} + D_{dn}(q)/B_{dn}$. Suppose the mobile phone consumes P_d watts for the data exchange, and the associated energy consumption is $[D_{up}(q)/B_{up} + D_{dn}(q)/B_{dn}] \cdot P_d$.

On the overall, when q is offloaded to the remote server, the total energy consumption $E_r(q)$ for the mobile phone is

$$E_r(q) = s_r \cdot P_r \cdot R^2(q) + P_d \cdot [D_{up}(q)/B_{up} + D_{dn}(q)/B_{dn}] \quad (2)$$

Now let us discuss the energy saving $\triangle E(q)$ of the remote approach over the local approach as follows.

$$\begin{aligned} \triangle E(q) &= E_l(q) - E_r(q) \\ &= (s_l \cdot P_l - s_r \cdot P_r) \cdot R^2(q) - P_d \cdot [D_{up}(q)/B_{up} + D_{dn}(q)/B_{dn}] \end{aligned}$$

We then rewrite $\triangle E(q)$ by:

$$\triangle E(q) = [P_l - \frac{P_r}{s_l/s_r}] \cdot s_l \cdot R^2(q) - P_d \cdot [D_{up}(q)/B_{up} + D_{dn}(q)/B_{dn}] \quad (3)$$

The saved energy $\triangle E(q) > 0$ obviously depends on the tradeoff between the reduced energy for the eliminated query computation operations (i.e., $[P_l - \frac{P_r}{s_l/s_r}] \cdot s_l \cdot R^2(q)$) and the increased energy for the extra data exchange $P_d \cdot [D_{up}(q)/B_{up} + D_{dn}(q)/B_{dn}]$. Obviously, for the given mobile phone and remote server, offloading q is more beneficial when large amounts of query computation operations $R(q)$ are needed yet with few amounts D_{up} and D_{dn} of data exchange.

3.3 Energy for the Hybrid Approach

We first give an observation of Eq. (2) as follows.

- The first subitem $s_r \cdot P_r \cdot R^2(q)$ is the energy for being idle, without contribution to the query q;
- The second subitem $P_d \cdot [D_{up}(q)/B_{up} + D_{dn}(q)/B_{dn}]$ is caused by the data exchange between the mobile phone and remote server, and benefits from a smaller value of $D_{up}(q)$ and $D_{dn}(q)$.

Based on the above observations, we give the basic idea of the hybrid offloading approach as follows. We denote that q_s is a subset of the $|q|$ terms of q and is offloaded to the remote server, and then $q \setminus q_s$ is the subset of the remaining terms and is processed by the mobile phone. Following Lemma 1 in Section 2.1, we then find that the query results of q must be the *superset* for the query results of q_s. Therefore, if the mobile phone offloads q_s to the remote server, it can practically reduce the amount of both $D_{up}(q)$ and $D_{dn}(q)$, and then reduce $P_d \cdot [D_{up}(q)/B_{up} + D_{dn}(q)/B_{dn}]$. Meanwhile, instead of being idle, the mobile phone answers the query results of $q \setminus q_s$.

Now the energy for the hybrid offloading approach, denoted by $E_h(q)$, contains two parts as follows.

- The first part is for the mobile phone to answer $q \setminus q_s$. Suppose that the mobile phones needs $R(q \setminus q_s)$ computation operations to answer $q \setminus q_s$. Similar to Eq. (1), the energy consumption for the mobile phone to answer $q \setminus q_s$ is $s_l \cdot P_l \cdot R^2(q \setminus q_s)$.
- Next, the second part is for the data exchange of the query results of q_s. Denote $D_{up}(q_s)$ and $D_{dn}(q_s)$ to the data size of the subset query q_s and the query results of q_s, respectively. Then, the associated energy consumption of the data exchange for q_s is $P_d \cdot [D_{up}(q_s)/B_{up} + D_{dn}(q_s)/B_{dn}]$.

Now, $E_h(q)$ is given as follows [1].

$$E_h(q) = s_l \cdot P_l \cdot R^2(q \setminus q_s) + P_d \cdot [D_{up}(q_s)/B_{up} + D_{dn}(q_s)/B_{dn}] \quad (4)$$

We now show that energy of the hybrid solution given by Eq. (4) consumes less energy than energy of the local and remote approaches given by Eqs. (2-3): Due to $q_s \sqsubseteq q$, the query results of q_s must be the subset of the query results of q (by Property 1). Thus, the claim of $D_{up}(q_s) \leq D_{up}(q)$, $D_{dn}(q_s) \leq D_{dn}(q)$ and $E_h(q) \leq E_r(q)$ surely holds. Consequently, we ensure that $E_h(q)$ in Eq. (4) is smaller than $E_r(q)$ in Eq. (2).

Summary: Essentially, the hybrid approach can be treated as a general solution for the above remote approach (i.e., $q_s = q$ and $q \setminus q_s = \emptyset$) and local approach (i.e., $q_s = \emptyset$ and $q \setminus q_s = q$). In view of this, to enable the least energy consumption for $E_h(q)$, the key is to select the subset query q_s from the query q. We denote SSQ to be the problem of selecting the subset query q_s in order to minimize the energy $E_h(q)$ of Eq. (4). In the next section, we solve the SSQ problems by using the min-cost flow approach.

4. MIN-COST FLOW-BASED SSQ

In this section, we first give an overview of the min-cost flow problem, then reduce the SSQ to the min-cost flow problem, and finally propose the practically useful optimization techniques.

4.1 Min-cost flow

We choose to represent the selection of q_s using a standard flow network. The flow network is a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V}$ and $\mathcal{E}$ are the sets of all vertexes and edges, respectively. In the graph, each edge $e \in \mathcal{E}$ is annotated with a non-negative capacity y_e and a cost s_e, and each node $v \in \mathcal{V}$ is annotated with a non-negative supply p_v with $\sum_{v \in \mathcal{V}} p_v = 0$. A feasible flow assigns a non-negative flow f_e ($\leq y_e$) to each edge e, such that for every vertex v, the condition $p_v + \sum_{e \in \mathcal{I}_v} f_e = \sum_{e \in \mathcal{O}_v} f_e$ holds, where $\mathcal{I}_v$ (resp. $\mathcal{O}_v$) is the set of incoming edges to v (resp. outgoing edges from v). In a feasible flow, $p_v + \sum_{e \in \mathcal{I}_v} f_e$ is referred to as the flow through the node v.

A *min-cost* feasible flow is a feasible flow that satisfies the following optimization problem:

[1]We assume that the time of the mobile phone to process $q \setminus q_s$ is larger than the time of being idle. Otherwise, if $q \setminus q_s$ contains few terms and q_s is processed by remote servers, it obviously favors using remote servers to process q.

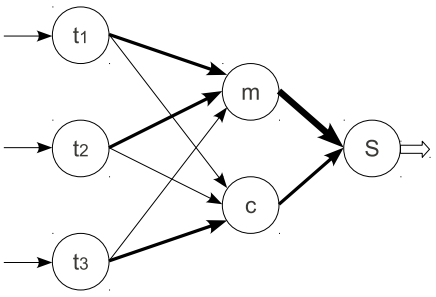


Figure 1: Min-cost flow-based term selection

Minimize	$\sum_{e \in \mathcal{E}} (s_e \cdot f_e)$
Subject to	(1) for each edge e, $f_e \leq y_e$;
	(2) for each node v, $p_v + \sum_{e \in I_v} f_e = \sum_{e \in O_v} f_e$;

For a graph with E edges and V vertexes, known worst-case complexity bounds on the min-cost flow problem are $O(E \log(V)(E + V \log V))$ and $O(VE \log(VP) \log(V^2/E))$, where P is largest absolute value of an edge cost.

4.2 Reducing SSQ to a min-cost flow problem

For simplicity, we first construct the min-cost flow graph G with a single query q and then extend the flow graph with N queries. For a single query q consisting of $|q|$ terms t_i ($1 \leq i \leq |q|$), the flow graph G is constructed as follows. Figure 1 shows the flow graph G and Table 2 summarizes the construction.

- Node set V: For each term $t_i \in q$ (with $1 \leq i \leq |q|$), we add a node t_i to G. For the mobile phone and remote server, we add two processing nodes m and c to G, respectively.
- Edge set E: For each term node t_i, we connect two directed edges from t_i to m and c, respectively.
- Single sink S: G contains a single sink node S, to which a directed edge starts from the nodes m and c, respectively.
- Capacity: each edge from the term node t_i to m (and to c) has the capacity 1, meaning that the term t_i is processed by m (and by c) at most one time. Also, the capacity of the edge from m (and the edge from c) to the sink node S is $|q|$, indicating that m (and c) processes at most $|q|$ terms.
- Supply: The nodes m and c have supply 0. Each term node t_i has a supply 1, and the sink node S has the supply $-|q|$, indicating S demands at most $|q|$ terms to process.
- Flow: (i) The edge from t_i to m (resp. to c) gets flow either 0 or 1 (resp. 1 or 0), depending on the solver of the SSQ problem. If the edge from t_i to m gets 1, then the term t_i is locally processed and there is no flow from t_i to c. Similarly, if the edge from t_i to c gets 1, then the term t_i is offloaded to the remote server and there is no flow from t_i to m. In addition, (ii) the edge from m (or from c) to S gets a flow zero or a positive number smaller than $|q|$, depending on the number of terms that m (or c) is processing.

	Capacity	Supply	Flow	Cost
term node t_i	nil	1	nil	nil
processing nodes: m, c	nil	0	nil	nil
sink node S	nil	$-\|q\|$	nil	nil
edge $t_i \rightarrow m$	1	nil	1 or 0	$s_l \cdot P_l \cdot R^2(t_i)$
edge $t_i \rightarrow c$	1	nil	1 or 0	$P_r[\frac{D_{up}(t_i)}{B_{up}} + \frac{D_{dn}(t_i)}{B_{dn}}]$
edge: $m \rightarrow S$, $c \rightarrow S$	$\|q\|$	nil	$[0, \|q\|]$	0

Table 2: Construction of the simple flow graph G

- Cost: On the edge from t_i to m, the cost per unit flow is $s_l \cdot P_l \cdot R^2(t_i)$, where $R(t_i)$ is the number of computation operations to answer the subset query consisting of a unique term t_i; on the edge from t_i to c, the cost per unit flow is $P_r[\frac{D_{up}(t_i)}{B_{up}} + \frac{D_{dn}(t_i)}{B_{dn}}]$. Instead on the edge from m (and c) to S, the cost is 0.

After the graph G is constructed, existing solvers can find the min-cost flow by polynomial time (as shown in Section 4.1). For each term node t_i, its supply (i.e., equal to 1) is injected to two outgoing edges (each of which has the capacity 1), the flow from the node t_i must go through the edge with the lowest cost in order to minimize the overall cost. In addition, considering that some documents contain multiple query terms, we further optimize the above model by reducing the corresponding cost if such terms are assigned to the same node (either m or c).

We use Figure 1 as an example. Suppose that a solver of the min-cost flow graph in this figure respectively injects the flow of 1 unit on the edges of $t_1 \rightarrow m$, $t_2 \rightarrow m$ and $t_3 \rightarrow c$, the flow of 2 units on $m \rightarrow S$, and the flow of 1 unit on $c \rightarrow S$. These flows indicate that the subset q_s (having two terms $\{t_1, t_2\}$) is offloaded to the remote server and the subset $q \setminus q_s = \{t_3\}$ is processed by the mobile phone.

4.3 The SSQ problem with N queries

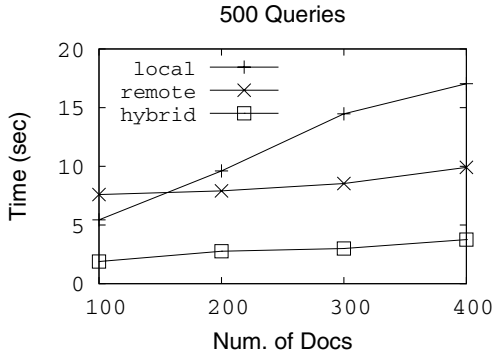
Now let us consider the SSQ with N queries as follows. For N queries, we assume there exist the totally T terms t_i ($1 \leq i \leq T$), and the frequency of t_i is T_i (i.e., the term t_i appears in T_i queries).

Similar to the construction of G, the flow graph G' for N queries consists of T term nodes, two processing nodes (m and c) and the sink node S, together with the associated directed edges. In addition, the supply of the nodes and capacity of the edges in G' are the same as the ones in the flow graph G in Section 4.2. However, the cost per unit flow of two graphs differ. In detail, the cost per unit flow on the edges $t_i \rightarrow m$ and $t_i \rightarrow c$ in G' is $T_i \cdot s_i \cdot p_i \cdot R^2(t_i)$ and $T_i \cdot P_d[\frac{D_{up}(t_i)}{B_{up}} + \frac{D_{dn}(t_i)}{B_{dn}}]$, respectively. The cost in G' is T_i times of the cost in the graph G. After the graph G' is constructed, we can similarly solve the min-cost flow problem to optimally find the minimal energy consumption.

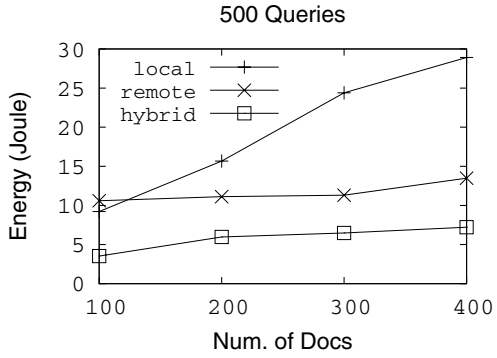
5. PRELIMINARY EVALUATION

We first report the used datasets, and then present our preliminary evaluation results.

(1) Query Logs: We use an input query history file (containing 4,000,000 pre-processed queries) collected from the Microsoft *MSN* search engine (*MSN* in short). Among such queries, we randomly choose 500 queries, and submit such chosen queries from mobile devices. On average, the number of terms per query is 2.14. Furthermore, the cumulative percentage of all queries containing at most 1, 2 and 3 terms is 34.8%, 71.2%, and 88.4%, respectively. These numbers, consistent with the previous work [17], indicate that real users prefer to use short queries, on average only 2 – 3



(*a*) Latency



(*b*) Energy

Figure 2: Preliminary Results

terms. The short queries lead to the flow graph G' having fewer term vertices and smaller computation cost.

(2) Content Documents: We use the standard Text Retrieval Conference (TREC) AP data set, a text categorization task based on the Associated Press articles used in the NIST TREC evaluations [2]. We randomly select 400 articles from the TRACE AP set, with on average 6054.9 terms per article. We pre-process the data set with the Porter algorithm [2] and remove common stop words such as "the", "and", etc.

(3) Experimental Methodologies: We use a Google Nexus S mobile phone with Android 2.3 version (1GHz Cortex A8 CPU, 512MB Ram) and a MacBook Air Notebook with Mac OS X OS (Intel Core 2 Duo 1.86 GHz CPU and 4 GB RAM) as the server to set the experiment testing environment. The phone access the server via a WiFi connection with 500KB/s upstream and 1MB/s downstream bandwidth. In our preliminary experiment, both the remote server and mobile phone maintain the same documents, and index the documents by main memory-based inverted lists. We initiate 500 queries from the mobile phone, and test the total time and energy for such queries by using three approaches: the local approach, remote approach, and hybrid approach.

(4) Experimental Results: Figures 2 (a-b) plot the time and energy for the three approaches.

First, when the number of documents is increased, these approaches have to spend more time and energy to retrieve the matching results.

Second, when the number of documents is smaller than 200, e.g., only 100 documents, the remote approach spends the largest time and energy, and the hybrid approach uses the least time and energy. It is because given a small number of documents, the retrieval of the inverted list from main memory is cheaper than waiting for the results from the remote server.

Instead, given more than 200 documents, the local approach uses the largest time and energy, and the hybrid approach uses the least time and energy. It is because the index size for such documents is large. Given the memory limit, in the local approach, the mobile phone spends more efforts to load the index into the main memory and retrieve the matching results over the index, incurring more time and energy than the remote approach.

In terms of the hybrid approach, it adaptively decides the locations to process the terms, and thus has the least time and energy. For example, given 500 documents, the energy of the hybrid approach is only the rate of 0.25 and 0.53 of energy of the local and remote approaches, respectively.

6. RELATED WORK

Years ago, Virpi Roto [14] had already found that similar to traditional Web search via personal computers, users are surprisingly willing to use search also with the traditional phone keypad. In order to study the difference between the mobile search and the traditional Web search, [17] reported the characteristics of search queries submitted from mobile devices using various Yahoo! oneSearch applications during a 2 months period in the second half of 2007. It gave some interesting results in terms of the mobile query patterns, such as on average 2-3 words per query, and only 13% queries having the frequency 1 (meaning such queries are submitted by users by only one time, and all other queries are repeated by more than 1 time).

Microsearch [15] built a search system suitable for small embedded devices used in ubiquitous computing environments (e.g., PDA). It indexed information stored within such a small device, and returned a ranked list of possible answers in response to a user's query (consisting of input terms). Snoogle [16] was an information retrieval system on low-cost wireless sensor networks, and served as the search engine and helped people to search physical objects at their vicinity. Microsearch allowed users to do textual search in the local storage of a stand-alone small device, yet Snoogle answered the keyword searches over the content objects stored in distributed sensors.

Recently, to save energy consumption of mobile phones, offloading tasks from the mobile phones to remote powerful side is frequently exploited in literature. For example, [9] examined the scenarios that would benefit from offloading, and draw the rule of thumb that the task must require more than 1000 cycles of computation on Nokia N810 and N900 device for each transferred byte of data to be beneficial. [5] considered the energy aspect and developed equations to see when offloading is beneficial. In addition, the recent work [6] analyzed the trade-off between local and remote processing the keyword search from the viewpoint of energy use.

The above works either answered the queries by the mobile side (e.g., [15]), or offloaded queries to the server side (e.g., [9, 5, 6]). Instead, we explore the possibility of adaptively answering the queries by a hybrid approach.

In addition, the previous works [11, 12, 10, 13] were focusing on keywords-based content filtering and dissemination (intuitively treated as a reverse application of keyword searches) on large distributed systems such as P2P networks and NoSql Cloud platforms. Significantly differing from such works, in this paper, we target on mobile phones to save energy for keyword searches.

Finally, [8, 4] proposed a fine-grained runtime offloading system,

called adaptive infrastructure for distributed execution (AIDE), and an offloading inference engine (OLIE), for resource-constrained mobile devices. They focused on when to timely trigger offloading and what partitioning policy used to select objects for offloading. [3] presented a middleware platform that can automatically distribute different layers of an application between a mobile phone and remote server, and optimized a variety of objective functions (latency, data transferred, cost, etc.). It modeled applications as a consumption graph, and proposed algorithms to find the optimal distribution of the application modules. These works run the partition algorithms by mobile phones. We focused on how to save energy for keyword searches, and in our work, the SSQ problem is solved by powerful servers instead of mobile phones.

7. CONCLUSION AND FUTURE WORK

In this paper, we studied energy-aware keyword searches on mobile phones and proposed three approaches. The proposed hybrid approach adaptively splits the keywords of queries into two subsets, such that one subset is answered locally by the mobile phones, and another is offloaded to a remote server. Our preliminary experimental results verify that the hybrid approach outperforms the two other extremes.

As future works, we are considering advanced query models (such as the threshold-based and top-k models). Moreover, as more general scenarios, where content and index files might not be consistent between mobile phones and the remote side, we are interested in how the hybrid solution correctly finds the needed content meanwhile at the cost of the least energy. Finally, with a large document collection, e.g., more than 10 GB of personal documents, the inverted list used to index such documents could be larger than the RAM size of mobile phones, and thus we consider the permanent storage implementation of the inverted list.

8. ACKNOWLEDGMENTS

This work is partially supported by Academy of Finland (Grant No. 139144), and we would like to thank the anonymous SIGCOMM 2012 MCC workshop reviewers for their valuable comments. Part of the research was also conducted in the Internet of Things program of Tivit (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT), funded by Tekes.

9. REFERENCES

[1] http://en.wikipedia.org/wiki/iphone.

[2] http://trec.nist.gov/data.html.

[3] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso. Calling the cloud: Enabling mobile phones as interfaces to cloud applications. In *Middleware*, pages 83–102, 2009.

[4] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. S. Milojicic. Adaptive offloading inference for delivering applications in pervasive computing environments. In *PerCom*, pages 107–114, 2003.

[5] K. Kumar and Y.-H. Lu. Cloud computing for mobile users: Can offloading computation save energy? *Computer*, 43:51–56, 2010.

[6] E. Lagerspetz and S. Tarkoma. Mobile search and the cloud: The benefits of offloading. In *PerCom Workshops*, pages 117–122, 2011.

[7] H. Liu, V. Ramasubramanian, and E. G. Sirer. Client behavior and feed characteristics of rss, a publish-subscribe system for web micronews. In *Internet Measurment Conference*, pages 29–34, 2005.

[8] A. Messer, I. Greenberg, P. Bernadat, D. S. Milojicic, D. Chen, T. J. Giuli, and X. Gu. Towards a distributed platform for resource-constrained devices. In *ICDCS*, pages 43–51, 2002.

[9] A. P. Miettinen and J. K. Nurminen. Energy efficiency of mobile clients in cloud computing. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, HotCloud'10, pages 4–4, Berkeley, CA, USA, 2010. USENIX Association.

[10] W. Rao, L. Chen, and A. W.-C. Fu. STAIRS: Towards efficient full-text filtering and dissemination in DHT environments. *The VLDB Journal*, pages 1–25, 2011. 10.1007/s00778-011-0224-z.

[11] W. Rao, L. Chen, P. Hui, and S. Tarkoma. Move: A large scale keyword-based content filtering and dissemination system. In *ICDCS*, 2012.

[12] W. Rao, A. W.-C. Fu, L. Chen, and H. Chen. STAIRS: Towards efficient full-text filtering and dissemination in a DHT environment. In *ICDE*, 2009.

[13] W. Rao, R. Vitenberg, and S. Tarkoma. Towards optimal keyword-based content dissemination in dht-based p2p networks. In *Peer-to-Peer Computing (P2P)*, pages 102–111, 2011.

[14] V. Roto. Search on mobile phones. *JASIST*, 57(6):834–837, 2006.

[15] C. C. Tan, B. Sheng, H. Wang, and Q. Li. Microsearch: A search engine for embedded devices used in pervasive computing. *ACM Trans. Embedded Comput. Syst.*, 9(4), 2010.

[16] H. Wang, C. C. Tan, and Q. Li. Snoogle: A search engine for pervasive environments. *TPDS*, 21(8):1188–1202, 2010.

[17] J. Yi, F. Maghoul, and J. O. Pedersen. Deciphering mobile search patterns: a study of yahoo! mobile search queries. In *WWW*, pages 257–266, 2008.