SAW Workshop 2010 at HPCA-16

Smart Phones – Future Devices, Usages and Applications

Roy Want, Intel Labs
2200 Mission College Blvd
Santa Clara, CA 95054
roy.want@intel.com

Abstract: We are in the middle of a mobile computing revolution, perhaps as significant as the revolution in the 1980s which defined the personal computer. Today, smart phones are providing us with an unprecedented opportunity to innovate in terms of the platform design, its use, and the applications we expect it to support. This paper examines smart phone design opportunities around sensing to support context aware operation; composability, enabling resource sharing to overcome platform limitations; and VM migration, to optimize processing performance while maintaining seamless operation.

1. Introduction

The growth in cell phone unit sales over the last 10 years has been unprecedented. Even through the recent recession, the market can still support greater than 1B units/year in 2009. Most of these devices are relatively modest cell phones, but an increasingly large percentage are now classified as *smart phones*, combining regular voice communication with data services and computation in the handset. Gartner estimated that in the first quarter of 2009 the percentage of smart phones sold increased by 12.7% over last year to 36.4M units[1] making smart phones a ~144M unit business, and comparable to laptop computers in volume sales. However, compared to the uniformity of form and function of PCs, this is a very diverse market with a wide variety of designs, display sizes, and interaction techniques. Furthermore, the operating systems are also diverse, with the Android OS recently being added to the mix. In many respects this resembles the PC market before the introduction of the IBM PC when there were a large number of contenders for the market, but many of them falling by the wayside. IBM created a hardware platform packaged with an operating system that the industry could rally behind. Supporting open development, third party vendors jumped at the opportunity, developing a wide variety of software for office and business needs. When Microsoft Windows was introduced and later refined, the industry convergence was established around the PC platform as we know it today. This paper examines three opportunities that may help drive towards a similar convergence in the Smart Phone platform allowing users to capitalize on the core benefit of mobility without compromising functionality.

The first is context awareness [5], enabled by embedded sensing technologies in the platform [2]. In this respect the introduction of the Apple iPhone was a significant development for the industry. It pushed the boundaries of design, making use of sensors to support novel interaction techniques. For example, the multi-touch display for selection and zooming; and various context sensors, such as an accelerometer, for gaming and navigation; a light sensor, to dim the display and save power when placed to your ear; and the use of received signal-strength indication (RSSI) from cell towers, to support location-based services (LBS). The iPhone also dispelled the myth that it was not possible to support an adequate web browsing experience on a mobile phone. In fact it set a new standard for mobile web-browsing, and today we see many copy-cat solutions. An observation supported by the iPhone success is that novel sensors can be used to radically change how a user interacts with a mobile device. The ability to seamlessly integrate new sensors into a standard platform is therefore a key enabler for rapid innovation (section 2).

The second is the ability to use a high-bandwidth wireless local-area-network (WLAN) to share nearby resources. In general terms, the smart phone is always going to be a compromise in design over a resource rich platform, such as a laptop or desktop computer. Despite innovations in smart phone interaction techniques, for many real world tasks, users still need more comprehensive access to computation and media than is available on the platform. A system that can be used to overcome this problem is called Dynamic Composable Computing [7], allowing mobile devices to share the computing resources of other nearby computers, or computing infrastructure, using a wireless network. For example, accessing a large display situated close-by is a more effective way of viewing a business document, rather than using a screen that is only a few inches wide. Even if we increase the density of pixels in the phone display it does not help very much because we are soon limited by the resolution of our eyes. Thus wireless composition (section 3) has the potential to play an important role for improving the usability of a smart phone.

The third is a capability to support seamless migration of computation between proximate computers [6]. When smart phones are able to support a standard computation environment, users will expect some level of processing continuity as they work on different platforms (e.g., home PC, mobile, car, office PC). One way to achieve continuity of task is to migrate computation across a WLAN between these computers by transferring a virtual machine (VM) image. We discuss the requirements to support VM migration in section 4. Finally, we conclude with a summary of smart phone platform design opportunities in section 5.

2. Always-on Sensing

Adding sensors to a mobile platform creates a number of design challenges. First, there are no consistent methods for interfacing physical sensors, and an engineer is faced with a collection of industry standards that included I2C, SPI, UART, along with analog and custom digital solutions. Furthermore, to facilitate development, these devices should be presented to a programmer using a consistent API. Although many commercial mobile devices have already incorporated some form of sensing, it has mostly been done in an ad hoc fashion, and it is generally not extensible. The second problem is that some mobile applications require sensor data to be always available, and thus results in a power management challenge to preserve smart phone battery life.

One approach for rationalizing how sensors are presented to a smart-phone host-processor is to design a sensor-hub that abstracts away the various hardware solutions into a single uniform interface. A secondary low-power core (or separate microcontroller) is an effective way to do this, as it allows for firmware upgrades to enable extensibility. The sensor hub can also serve as a preprocessor for recorded data, providing buffering, DSP and threshold-based reporting to the host processor. The latter also allows the host to sleep and save power in the absence of critical data. We can summarize the requirements of a sensor hub as follows:

- Command & Reply format: a consistent control syntax between host & hub
- Streaming: a periodic data channel from the hub that returns sensor data to the host
- Errors: a mechanism to report when a command fault occurs, or when a stream cannot meet its time reporting commitments.
- Timestamps: labeling all sensor events using a common time reference
- Buffering: enabling sensor data to be collected until ready for burst transfer
- Preprocessing: enabling digital signal processing of the data e.g. filtering
- Thresholds: data is reported only when it exceeds or falls below defined thresholds
- Power monitoring: reporting the power usage of a sensor(s) to the host.

In order to experiment with this functionality, and demonstrate the utility of this approach, we have designed and implemented a sensor hub with these characteristics (Figure 1)

This hub has been designed with a wide variety of sensors to enable general experimentation (3D-accelerometer, 3D-gyro, 3D-compass, barometer, lux meter, thermometer, multi-channel touch input, NFC, IrDA, ultrasonic ranging, a 2D optical blob-tracker, and mechanical switches). It connects to a host through a USB interface providing power and a bidirectional data channel.

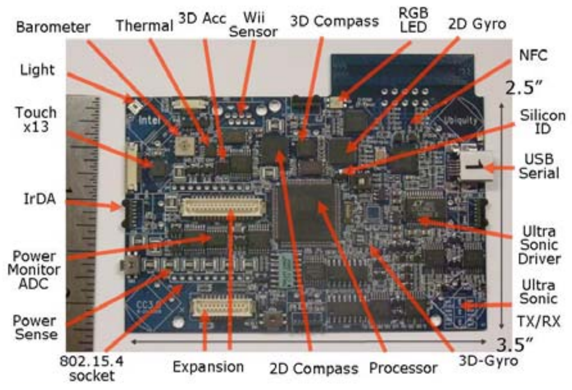

Figure 1: Experimental Senor-Hub with USB/serial interface

To allow for design extensibility, and a consistent API, we chose a textual interface using key/value pairs e.g., the 3D accelerometer could be turned on using the following text command:

```
acc op=stream val=on period=1000 power=on thresh-high=100:100:100
```

In this command the streaming operation is turned on with period 1000mS, and power monitoring also enabled. The high-threshold for reporting is set to 100 for the X, Y and Z dimensions. A periodic response string will be returned assuming any of X, Y, and Z cross the threshold, resulting in a periodic event string, for example:

```
EVT:0987893765  acc val=120:50:-10 p=22 s=1
```

A string beginning with EVT tells the host it is a periodic event occurring at the system clock time which follows the colon. The X, Y, Z reading of the accelerometer are contained in the 'val' string each separated by colons. The 'p' parameter is the power in milliwatts (22mW) and 's' is the sequence number, in this case the first reported event of the series.

This command syntax allows more key/value pairs to be added as new versions of the system are developed, providing the required extensibility for both commands, and the data responses issued by the hub. However, if the bandwidth of the serial link is very limited, a more compressed binary encoding of the response parameters can be used instead of a text encoding.

The string command syntax for each of the sensor types is identical except for the key word used to identify it. The hub thus makes programming extremely easy and hides the differences and underlying complexity of each sensor interface.

As the cell phone industry converges on designs based on a System-On-Chip (SOC) methodology, a key future requirement will be a sensor-hub subsystem within the SOC to provide functionality similar to that described. If future mobile silicon solutions support multiple asymmetric cores in the SOC, one of the low-power cores could be dedicated to the function of the sensor-hub.

3. Sharing Nearby Resources

Any mobile platform results in a design compromise between its expected use and the engineering constraints, such as the display resolution, keyboard size, processing power, storage, enclosure dimensions and battery life. As a result, a smart phone is likely to have many more limitations compared to a larger less mobile device such as a desktop computer. Since the year 1999, standard wireless LAN technologies have provided a solution. WiFi with its ever increasing bandwidths, demonstrated by the progression of 802.11b (11Mbps) to 802.11n (400Mbps), now allows proximate computers to transfer data at rates approaching what used to be the domain of wired networks. As a result it is possible to take advantage of IP resource sharing solutions that were originally designed for inter-operation across the Internet. Examples of resources that can be shared in this way include displays, storage devices, network adapters, USB peripherals and sensors. The computer industry has already created standards for sharing many of these devices.

For example, virtual network computing (VNC) allows a computer running a VNC-server to share its screen with a second computer running a VNC-client. It does this by monitoring the frame buffer of the source computer, and sending compressed representations of any changes to the destination client, thus keeping network utilization to a minimum. Likewise, keyboard and mouse events at the client are sent back to the server where local keyboard and mouse events are simulated. Thus users can access applications running on a remote computer, but have a similar experience as if working on a local computer.

Similarly the SMB protocol used by the Samba server allows the sharing of a disk across a network. In the case of a Windows PC, a network drive can be mapped as a new drive letter and as far as a computer application is concerned, it appears to as just another locally attached disk. There are also commercial software solutions for sharing USB devices such as *Redirector*. Network interface sharing can also be achieved adapting standard IP routing tables. Sharing sensing over an IP network requires more customized solutions, but in many cases can be piggybacked onto existing mechanisms for sharing keyboard events. Thus most of the building blocks for resource sharing already exist.

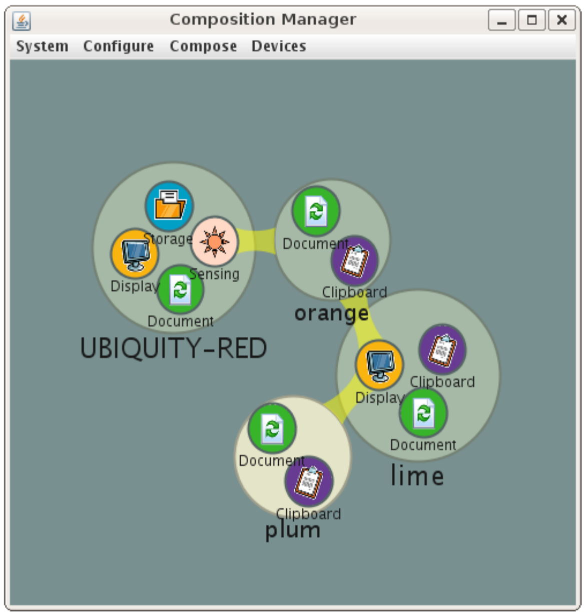

Figure 2: The Composition Manager: showing four nearby computers that have discovered each other (large circles), and the resources they can share (small circles). The status of resource sharing is shown by yellow lines.

An important characteristic of these sharing mechanisms is that legacy applications will still continue to operate as before because they cannot tell which of their resources are in fact remote. This is one of the main reasons these protocols have been successfully adopted throughout the industry. From the perspective of wireless sharing of nearby resources, these protocols become even more effective because any latency or bottlenecks typically introduced, by a long-haul network are minimized in a one-hop high-bandwidth WLAN.

From a user perspective, an integrated approach for sharing resources is more desirable than a set of isolated tools. It also provides a general mechanism to build a logical computer from remote resources on different computers. Rather than requiring a user to initiate multiple commands to assemble all the resources, the system can provide a user with a single command to simplify the process. We call this system Dynamic Composable Computing (DCC) [7], and have built an implementation to support it, the *Composition Framework*. Within this framework computers not only discover each other, but the resources they wish to share. The discovered computers and their services are represented graphically in a program called the *Composition Manager*, and presented as a GUI to the user (Figure 2). The metaphor for sharing resources is designed to be simple, and initiated by drawing a line between a computer (the larger circle), and a smaller circle representing a resource. To release a resource, a stroke is drawn across the connecting line causing it to disappear.

The Composition Framework is implemented as a distributed program that must run on each of the computers that wish to share resources [3]. In addition to providing the uniform graphical interface we described above, it also allows symmetry of sharing because a user no longer needs to be concerned if they are using the client or server in the system. For example, a computer desktop can be pushed to a remote computer for display, or pulled from the remote computer to display it locally, using the same graphical interface. The device and service discovery mechanism is achieved using IP broadcast, and assumes each computer is already connected to the same subnet e.g. associated with the same wireless access point. However, it could also use Layer-2 discovery if an ad-hoc wireless technology is available such as the new WiFi PAN standard, WiFi-Direct.

From the point of view of the Composition Framework, the system is agnostic to the underlying resource sharing protocol. This is because the client and server components for each protocol are scripted in a series of configuration files, with the appropriate arguments appended to invoke the appropriate command. As more efficient sharing solutions are designed, possibly with hardware support, the Composition Framework configuration files can be simply updated with minimal effort. In terms of future SOC designs, wireless display sharing is a prime candidate for hardware acceleration, presented to the Composition Framework as a client/server software wrapper which can then be scripted by the system.

4. Augmented & Seamless Processing

Using nearby high-performance computers to augment processing on a small mobile device is a form of resource sharing, going beyond system peripherals to sharing the processor itself. However, unlike peripheral sharing mechanisms such as VNC, this has not been widely explored for mobile computing.

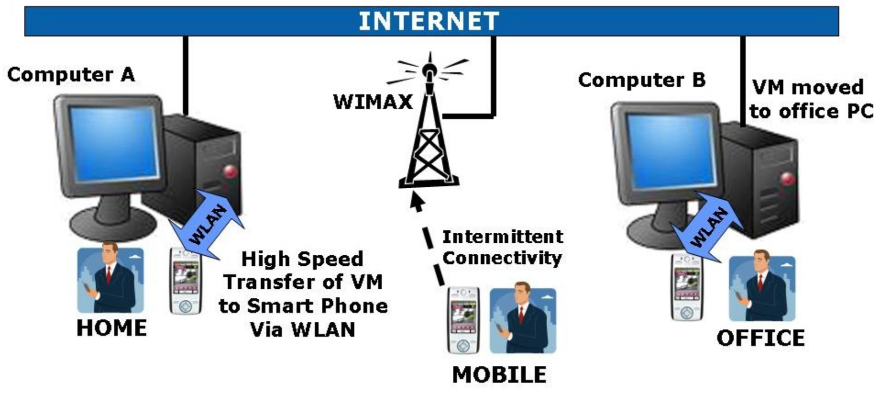

Figure 3: A Smart Phone migrating computation between Home & Office PC using a VM

One of the most promising approaches to enable this capability is in the realm of virtual machine migration, because it has only recently been practical between mobile and desktop machines with the advent of low-power mobile x86 processors. The Intel ATOM x86 family of processors was announced in April 2008, and have now been shipping for just over a year in the form of Mobile Internet Devices (MIDs) and Netbooks, which also run standard Windows and Linux based operating systems. The next generation of ATOM processors will target Smart Phones, and run x86 applications without modification. The resulting shared instruction set allows the same code to run on ATOM-based mobile computers as it does on most desktop computers, and thus standard Virtual Machine (VM) hypervisors, and the associated VM migration techniques, can be employed to support the migration of computation between the two.

Consider the everyday scenario (Figure 3) of a mobile professional starting out at home working on a PC in the den, and then running some errands in town while periodically sending emails and accessing key documents on her mobile computer. In order to provide a seamless transition her entire computational state can be suspended in a VM image on the PC, and then transferred to the mobile where it can be resumed. Likewise, when she arrives at her office workplace, her mobile can also suspend its VM state and transfer it to the office PC where it can resume and continue on a computer with much higher performance.

ATOM based smart phones makes this mobile use model a reality. A collaboration between the University of Toronto and CMU have developed the Horatio system [6] expanding the earlier work on Internet Suspend Resume (ISR) [4] to include migration to a smart phone. This system demonstrates the opportunity of using either a network server, or a mobile device, to transfer a VM image to a secondary PC. However, the first version of the Horatio system used ARM based smart phones and therefore could not resume the VM image while mobile. In a few years, x86 smart phones will become more common, thus enabling the more complete use model.

One of the challenges for VM migration is the potentially large amount of data (typically 4-20GB) that needs to be transferred to successfully move a suspended VM image. Higher bandwidth WLANs make this more practical because the transfer time is reduced. ISR employs a differencing technique to transfer the image, comparing the hashes of disk blocks from a previously transferred VM image with those on the source computer, and then only transferring blocks with differing hash values. For typical image deltas, due to changes that result from standard workloads, an actual transfer of only 50-100MB of data may be required. These techniques are however based on software solutions, and VM image transfers can be accelerated with hardware. An SOC solution could automatically calculate disk-block hash values, and orchestrate the transfer of disk blocks using data compression across the WLAN directly into the file system of the target machine. Hardware supported encryption/decryption of disk blocks is also important for migration between untrusted machines, and hardware based encryption can also accelerate this process.

5. Conclusion

Modern smart phone designs will make use of sensing, to support context-aware operation; composable computing protocols, to support peripheral sharing; and VM migration between x86 processors, to augment mobile processing. Although all three capabilities can be demonstrated using existing architectures with software support, for optimal user experience & power efficiency, platform-level hardware support in an SOC is a desirable addition to the platform.

Acknowledgments: Trevor Pering, Kent Lyons, Shivani Sud, Barbara Rosario, Michelle Gong and Alex Nguyen for their contributions to the systems and observations described in this paper.

References

[1] Mobile Phone Sales Q1 2009: Gartner: http://www.gartner.com/it/page.jsp?id=985912
[2]. S. Kang, J. Lee, H. Jang, H. Lee, Y.Lee, Taiwoo Park, J. Song. "SeeMon: Scalable and Energy-efficient Context Monitoring Framework for Sensor-rich Mobile Environments". Proc. ACM MobiSys'09. Breckenridge, Colorado. June 2009
[3] T. Pering, R. Want, B. Rosario, S. Sud, & K. Lyons "Enabling Pervasive Collaboration with Platform Composition", Pervasive 2009, Nara , Japan May 11-14, 2009.
[4] M. Satyanarayanan, B. Gilbert, M. Toups, N.Tolia, A. Surie, D. R. O'Hallaron, A. Wolbach, J. Harkes, Adrian Perrig, David J. Farber, Michael A. Kozuch, Casey J. Helfrich, Partho Nath, H. Andrés Lagar-Cavilla. *Pervasive Personal Computing in an Internet Suspend/Resume System*, IEEE Internet Computing, Vol. 11, No. 2, March/April 2007.
[5] B. Schilit, N. Adams, R. Want, "*Context-Aware Computing Applications*", Workshop on Mobile Computing Systems and Applications (WMCSA) , Dec 1994, Santa Cruz
[6] S. Smaldone, B. Gilbert, N. Bila, L. Iftode, E. de Lara, M. Satyanarayanan, "*Leveraging smart phones to reduce mobility footprints*", Proc. ACM MobiSys'09, 109-122. Krakow, Poland. June'09
[7] R. Want. T. Pering, S. Sud, and B. Rosario, "Dynamic Composable Computing" for ACM HotMobile 2008, February 2008, Napa Valley, California.