**Biography:**

Weichih Huang received his B.S. (2004) degree in National Taiwan Normal University and MS (2006) degree in National Tsing Hua University. He was a software engineer in Billion, based in Taiwan, from 2007 to 2011. Currently he is a PhD student supervised by Dr. William Knottenbelt at Imperial College London. His research interests include self-adaptive software systems and embedded systems.

**Title:**

Self-Adaptive Containers: Building Resource-Efficient Applications with Low Programmer Overhead

**Abstract**

As the variety of execution environments and application contexts increases exponentially, modern software is often repeatedly refactored to meet ever-changing non-functional requirements. Although programmer effort can be reduced through the use of standardised libraries, software adjustment for scalability, reliability and performance remains a time-consuming and manual job that requires high levels of expertise. Previous research has proposed three broad classes of techniques to overcome these difficulties in specific application domains: probabilistic techniques, out of core storage, and parallelism. However, due to limited cross-pollination of knowledge between domains, the same or very similar techniques have been reinvented all over again, and the application of techniques still requires manual effort. This talk introduces our vision of self-adaptive scalable resource-efficient software that is able to reconfigure itself with little other than programmer-specified Service Level Objectives and a description of the resource constraints of the current execution environment. Our approach is designed to be low-overhead from the programmer’s perspective – indeed a naïve implementation should suffice. To illustrate our vision, we have implemented in C++ a prototype library of self-adaptive containers, which dynamically adjust themselves to meet non-functional requirements at run time and which automatically deploy mitigating techniques when resource limits are reached. We describe the architecture of the library and the functionality of each component, as well as the process of self-adaptation. We explore the potential of our library in the context of a case study, which shows that the library can allow a naïve program to accept large-scale input and become resource-aware with very little programmer overhead.