En el año 2013, los centros de cálculo de los Estados Unidos consumieron el 2,2% del consumo total de electricidad en ese país. Las proyecciones futuras indican que esta cantidad se incrementará rápidamente durante la próxima década. Una cantidad significativa del consumo de un centro de cálculo corresponde al funcionamiento de los servidores, y un alto porcentaje de este consumo se desperdicia mientras los trabajos compiten en el uso de recursos compartidos. Una gran cantidad de los centros de cálculo se utilizan para ejecutar trabajos interactivos (como por ejemplo, servicios de búsquedas en la web, o aplicaciones de comercio electrónico), para los cuales es muy importante cumplir con las expectativas de los usuarios y proporcionar una alta calidad de servicio (CDS). En estos centros, se intentan ejecutar aplicaciones interactivas i en *batch* en la misma infraestructura para incrementar su utilización, y reducir los costes de mantenimiento y la energía total consumida. Aunque se dedican muchos esfuerzos al impacto de la compartición de recursos en el rendimiento de las aplicaciones, todavía se mantiene el problema de garantizar un determinado nivel de CDS para los dos tipos de trabajos, interactivos y en *batch*. Los objetivos de esta tesis doctoral son, en primer lugar, investigar mediante técnicas de modelado, cómo y hasta que punto la contención debida a la compartición de recursos tiene un efecto en la ejecución y el consumo en trabajos batch. En segundo lugar, la tesis presenta una técnica de planificación para determinar dinámicamente la mejor configuración para satisfacer una CDS en trabajos interactivos con un límite de latencia preestablecido, en cualquier arquitectura.

Para conseguir los objetivos propuestos, primero proponemos una técnica de modelización para estimar dinámicamente el rendimiento y el consumo de los servidores, que recibe por nombre Runtime Estimation of Performance and Power (**REPP**). El objetivo que perseguimos con la política de planificación REPP es permitir a los administradores obtener el control del consumo y el rendimiento de los procesadores. REPP consigue este objetivo a través de la estimación del rendimiento de las aplicaciones y su consumo al variar los niveles de energía del procesador (a través del cambio de frecuencia y voltaje – DVFS, del uso de estados de inactividad y de la parada de los propios procesadores), y dinámicamente cambia la configuración del sistema respetando las condiciones dadas por el usuario. Este modelado se realiza en base a un conjunto de contadores de eventos del procesador, que se han seleccionado de forma que están disponibles en las arquitecturas más comunes, haciendo que REPP sea independiente de la arquitectura.

En este trabajo de tesis doctoral, también defendemos que los métodos tradicionales de modelado y las estrategias de planificación usadas en estos entornos, no son efectivas para trabajos interactivos. Para tratar correctamente a estos trabajos, proponemos **Hipster**, una política de planificación que combina una heurística y un algoritmo basado en aprendizaje por refuerzo. El objetivo que fijamos con Hipster es mejorar la eficiencia en el uso de los recursos, al mismo tiempo que se respeta la calidad de servicio data a los trabajos interactivos. Hipster consigue sus objetivos con la exploración del funcionamiento del sistema y la variación de la frecuencia y el voltaje de los procesadores.

Durante el desarrollo de esta tesis doctoral, hemos implementado REPP y Hipster en plataformas comerciales de 64 bits (Intel SandyBridge y AMD Phenom II X4 B97) y experimentales (ARM big.LITTLE Juno R1). Hemos obtenido resultados experimentales en estas plataformas y hemos demostrado que REPP realiza estimaciones de consumo y rendimiento de aplicaciones secuenciales y de trabajos formados por varias aplicaciones. El error medio en las arquitecturas Intel, AMD y ARM son, respectivamente, del 7,1%, 9,0% y 7,1% en la predicción del rendimiento, y del 8,1%, 6,5% y 6,0% en la predicción del consumo. De forma similar, demostramos que al

comparar Hipster con los trabajos previos, nuestro algoritmo mejora la calidad de servicio para el servicio de búsqueda en la *web* (Web-Search), entre el 80% y el 96 %, y para la aplicación Memcached del 92% al 99%, al tiempo que reduce el consumo de energía hasta el 18% en la arquitectura ARM.